May/June 1981 — Volume 1, Number 3 — $2.50 (USA) £1.20 (UK)

The magazine for Sinclair ZX80 users

SYNC

Graphics:
- Automatic Display Changes
- Graphics Tricks
- Hunt for Gold
- Walls and Dikes

Translating from Other Basics

Keyclick Generator

Adventures of Crash Cursor

Games:
- Hammurabi
- Black Hole
- Gauntlet
- Forest Treasure
- Mastermind

Inside the ZX80 Keyboard

Handling Character Strings

SYNTAX ZX80®

A PUBLICATION OF **THE HARVARD GROUP**

SYNTAX ZX80 is a brand-new monthly newsletter created just for you. We bring you news, reviews and forecasts of hardware, software and applications for your ZX80 or MicroAce, as well as technical details for circuit-builders. SYNTAX also provides a forum for users to share advice and problems about programs, vendors and topics of mutual interest. As more products become available, we'll bring you the ads and releases that keep you informed.

At SYNTAX we emphasize practicality. You can apply our suggestions even if you aren't sure at first why they work, because we give you complete instructions. Text is clear and easy to understand. SYNTAX readers already know about:

- An automatic phone-dialer they can put together in a few hours
- Syntactic Sums™ to check input for errors
- Printing characters four times normal size
- Programs to explore computer memory
- Cassette eavesdropping to locate files on tape and simplify loading
- How to build their own external additional RAM
- How to add an 8212 I/O chip to control external devices from their computers

And SYNTAX readers like what they get every month. Subscribers know they can depend on us.

> *After receiving only three issues of SYNTAX ZX80, I find that I anxiously await the next issue . . . keep up the good work!*
>
> Martin Irons
> Goshen, NY

> *Congratulations on the brass-tacks, down-to-earth approach of your newsletter. I'll be looking forward to future issues.*
>
> Otis Imboden
> Washington, DC

Many readers get their first issue and immediately order the back issues – more proof that they like what they see.

What's special about our publication? Just look through one issue. We work hard to bring you a quality newsletter. We strive to print useful programs of above-average accuracy. As any computer magazine editor can tell you, program listing accuracy is tough to achieve, but we boost our average with every issue. We test each program to make sure it works, it fits in the designated RAM, and it runs when you follow the directions. We print program listings in screen-image format to make it easier for you (it's sure not easier for us!) to enter programs accurately. We invented Syntactic Sum™ as an additional aid for you in getting error-free programs. With your subscription you also get access to hundreds of other readers, and our staff experts are available by phone to answer your questions or help you solve problems with your machine.

SYNTAX readers get every month:

- Latest news of Z80 hardware and software
- Programs to organize information, calculate, entertain, or instruct
- Do-it-yourself additions to the ZX80/MicroAce
- Clear explanations for beginners

To share the benefits of SYNTAX ZX80, just complete the coupon below and return it with your choice of payment. You will receive a year's subscription, 12 issues, for only $25 in US funds (plus $13 for foreign airmail if you live outside North America).

We are so sure you'll find SYNTAX useful that we promise to refund your entire subscription fee if you aren't satisfied. An unconditional guarantee – you can't lose. But if you're still skeptical, write for a free sample issue and see for yourself how SYNTAX can help you use and enjoy your ZX80 or MicroAce more.

Join the others who stretch the ZX80s and MicroAces to their utmost. Act now – as soon as we receive your coupon with payment, your first issue will be on its way. For faster service, phone your credit card order to 617/456-3661. Don't miss SYNTAX!

THE HARVARD GROUP
Bolton Road, Harvard, Mass. 01451

YES! Please send me 12 issues of SYNTAX for $25.

☐ My check for $25 is enclosed. ☐ Please charge my ☐ MasterCard ☐ Visa ☐ American Express ☐ Diner's Club account.

account number ____________________

exp. date ____________ bank number (MC only) ____________

signature ____________________

Name ____________ Title ____________

Organization ____________________

Address ____________________

City ____________ State ________ Zip ________

Phone No. (   ) ____________ Home Phone No. (   ) ____________

I own a ☐ Sinclair ZX80
☐ MicroAce computer.

S681

The magazine for Sinclair ZX80 users

SYNC

May/June 1981

Volume 1, Number 3

Staff

Index to Advertisers

Volume 1, Number 3

SYNC is published bi-monthly for $10.00 per year by Creative Computing, 39 E. Hanover Ave., Morris Plains, NJ 07950. Second class postage paid at Morris Plains, New Jersey 07950, and additional entry offices.

Postmaster: Send address changes to SYNC, P.O. Box 789-M, Morristown, NJ 07960.

Subscriptions in USA: 6 issues $10; 12 issues $18; 18 issues $24. UK and foreign airmail subscriptions: 6 issues £10; 12 issues £18; 18 issues £25. Call (800) 631-8112 toll-free (in NJ, 201-540-0445) to begin your subscription.

ZX81 Announced in Great Britain

Sinclair Research is now advertising the new ZX81 computer in British computer magazines. An improved version of the ZX80 computer, the ZX81 has been redesigned to incorporate a number of new features. The ZX80 reduced the number of integrated circuits to 21, but the ZX81 further reduces the number of chips to four by using a new custom-built chip that replaces 18 others. The 8k Basic ROM chip (now also available for the ZX80 as a drop-in replacement) gives the user the capacity to use decimals with 8 place accuracy, to work with log and trig functions (with their inverses), to plot graphs, and to make animated displays. A new 40 key keyboard expands the number of key words that can be entered by one key stroke, e.g., PEEK, POKE, SCROLL. This eliminates typing out these words. Twenty new graphics characters and 54 inverse video characters increase the graphics capabilities. Users have the choice

of two speed modes: "slow" and "fast" which is four times the "slow" mode and comparable to other personal computers. The slow mode eliminates screen flicker. Although the 1K RAM is the same as for the ZX80, it can be expanded by plugging in the new 16K memory unit. (So can the ZX80.)

The ZX81 will sell in Great Britain for £69.95, but it will not be available in the U.S. for the foreseeable future. Even if we colonials fly to Britain to buy it, the British version will not work with an American TV set. However, ZX80 users can have most of the capabilities (excluding the animated display) if they upgrade to the new 8K Basic ROM which is now available in the U.S. for $39.95 plus shipping from Sinclair Research (see Resources Column).

Sinclair has also announced that a 32 column printer will be available in the summer of 1981 for about £50. This will work with the 8K ROM machines.

Software Review...

Unfriendly Skies

David Lubar

Name: *Super ZX80 Invasion* (1K and 2K)
Type: Fantasy Game
System: Sinclair ZX80; MicroAce
Format: Cassette
Language: Basic
Summary: Best action game we have seen for the ZX80.
Price: $14.95 plus $1.50 shipping
Manufacturer: SOFTSYNC, INC.
P.O. Box 480
Murray Hill Station
New York, NY 10156

A cult has grown around the game of *Space Invaders*. Individuals with glazed eyes and pockets full of quarters have been known to haunt arcades for hours, sending countless rows of aliens to a laser death. Now, Sinclair owners can experience the same mania in their own homes. Using an active display to produce true animation, Softsync has given us *Super ZX80 Invasion* for the Sinclair. The tape comes with both 1K and 2K versions of the game. Let's start with the 1K program.

The player has a ship (or laser base, depending on your interpretation) at the bottom of the screen. The ship can be moved left or right using the arrow keys. The 0 or 9 key is used for shooting. Above the player, rows of aliens rain down missiles. The aliens move slowly across the screen, and the entire group moves closer to the player on each pass. If you shoot them all before being hit five times, you are rewarded with another screenful of aliens. That's basically it. The 1K version doesn't keep score, so you have to remember how many frames of aliens you have destroyed. Your ship contains a number telling you how many ships are left. When the number reaches zero, the game starts over. There are three skill levels available in the 1K version.

The program is fast, which introduces a problem. You have no chance to get set. As soon as it starts, the aliens are shooting at you. You can lose two or three ships before even touching a key. When your last ship is destroyed, there is no pause. The game starts again. If you are down to one ship and one alien, it can be hard to tell who hit who.

The 2K version does keep track of the number of frames completed. It also allows for fifty different skill levels, more aliens, and extended play for each frame completed. Unfortunately, there is no way to stop the game and change skill levels. Once it is running, you can only stop it by pulling the plug. To go to a different skill level, you have to reload the tape.

Despite these problems, the game is fun, assuming you aren't easily frustrated. It is probably the best Sinclair game to hit the market so far. The programmers have to be congratulated for putting so much into 1K of space. □

Handling Character Strings in the ZX80

Hasse Taube

From the first day I tried to operate my new ZX80, I have been looking for a way to simulate some of the character-string operations possible in other programming languages, but not in the ZX80 Basic. An idea mentioned by Michael Kirkland in *Personal Computer World*, February, 1981, on using USR(47) to obtain the address of the end of the variables provided me with the key to start my programming tricks in the ZX80.

Consider the following piece of code:

```
100 LET A$="ABCDEFGHIJKLMNOPQ"
200 LET A=USR(47)
```

Then A is the address of the byte after the last "".

Consider also:

```
100 LET A$="ABCDEFGHIJKLMNOPQ"
200 LET A=USR(47)-2
```

A will contain the address of the last byte of the previous character string; in this case, A will be the address of the letter Q.

In order to use this facility, you should not have any other statements between 100 and 200 in the examples mentioned above. To test this on your ZX80, enter the following short program:

```
100 LET A$="ABCDEFG"
200 LET A=USR(47)
300 PRINT CHR$ (PEEK(A))
```

This should give the letter G as output, i.e., the last character in a character string A$.

You know, of course, that you can always get the first character in a character string by a piece of code like this:

```
100 LET A$="ABCDEFGHIJ"
200 LET B$=CHR$(CODE(A$))
```

Then the variable B$ will get the value A, i.e., the first character of the character string A$.

Hasse Taube, Ericavej 39, DK 2820 Gentofte, Danmark.

It is, however, not so easy to get the last—or in fact any other than the first character. This is now possible with the use of the technique just described.

Suppose you want the last character of a string variable after a value has been assigned to it by an INPUT:

```
100 INPUT A$
200 LET A=USR(47)-2
300 PRINT CHR$(PEEK(A))
```

If you run this program and input, say, QWERTY, the output should be Y.

In other words, if you use:

```
(line number) LET A=USR(47)-2
```

immediately after an assignment of a string variable by a LET-statement or by an INPUT-statement, the variable A will point to the last character in the string.

If you know the length of the string, it is easy to take a substring from it. Suppose you know that the length of the string is 5 as in the following example:

```
100 LET A$="ABCDE"
200 LET A=USR(47)
300 LET B$="XX"
400 LET B=USR(47)
500 POKE B-2,PEEK(A-5)
600 POKE B-3,PEEK(A-6)
700 PRINT B$
```

Then the output will be AB, i.e., the first two characters from the string A$.

By using the example above with other values in statements 500 and 600, you could, of course, get another substring from A$. Also, if you would like to take a larger substring than just two characters as in the example, you probably would set up for a FOR...NEXT loop to do the POKE's and PEEK's.

Suppose you do not know the length of the string variable from which you want to take, say, the second and third characters. How can you find the length of a string variable? Several methods are available. The first uses the TL$ in a loop like this:

```
100 INPUT A$
200 LET B$=A$
300 FOR I = 1 TO 1000
400 LET A$ = TL$(A$)
500 IF A$ = "" THEN GO TO 1000
600 NEXT I
1000 LET A$=B$
1100 PRINT I
```

Then the output-value will be the length of the string which you input.

A much more interesting method for finding the length of a string also gives you the address of the beginning of the string. In this second method strings are internally stored in the ZX80 as follows:

—One byte with a code for the name of the string;

—The string itself from first character to last character;

—The ending quote.

The first byte contains a value which is equal to decimal 96 plus the code for a letter and which names the string. For example, a string named A$ will have 96+38=134 (decimal) in the first byte. A string named Z$ will have 96+63=159 (decimal) in the first byte. To understand these examples, you must know that the ZX80 representation for A is 38 and Z is 63. (See your instruction manual for the ZX80.)

To find the address of the first byte of the string in a string variable, you must set up a loop to test for the value in the first byte, described above. Assuming you know the address of the byte after the ending quote from USR(47), this should be fairly simple:

```
100 INPUT A$
200 LET A=USR(47)
300 FOR I=0 TO 1000
400 LET J=A-2-I
500 IF PEEK(J)=134 THEN GO TO 1000
600 NEXT I
1000 PRINT I
```

The program above will print out the length of the string which you input, but more interestingly, after statement 1000, J will point to the first byte of A$, and J+1 will point to the first byte in the string itself. □

More Truth in Programming

David Lubar

The other day, while blithely working on a program, I discovered that something was amiss. I had made an assumption about a certain Boolean operator, and passed on the information without checking my assumption. As Murphy's law would have it, the assumption was wrong. Going back to the article in issue two of *SYNC*, there is a discussion of various tricks with logical operators. For example, to test whether a number is not zero, you could use the following program.

```
10 INPUT N
20 IF N THEN PRINT "NOT ZERO"
```

There is no problem here. The expression will only be true if N has a value other than zero. I had assumed that the converse was also true, believing expressions such as

IF NOT N THEN PRINT "THE NUMBER IS ZERO"

would also work. To put it bluntly, they don't. The reason for this is that NOT can function in two different ways (actually, it always functions in the same way, but has two different applications). When working only with true and false (values of -1 and 0), NOT will always make a true expression false, and make a false expression true. So far, so good. As long as the universe is restricted to the values 0 and -1, there is no problem.

Before going on, try the following on your computer. Ask it to PRINT NOT (0). Then ask it to PRINT NOT (-1). As you can see, this works in the expected manner. Now try PRINT NOT (5). You might expect an answer of 0, since 5 (or any number other than zero) is considered to be true when evaluated logically. But life is not that simple. What NOT actually does is to take each bit in the byte and change it. Ones becomes zeroes, zeroes become ones. Now, if you've been trying all this, you'll have found that any positive number N, when used in PRINT NOT(N) will produce a negative number. This has to do with the way the Sinclair stores numbers. In positive numbers, the highest bit is set to zero. In negative numbers, this bit is set to one. Since NOT changes each bit, it will change the sign of most numbers.

You've probably also noticed that PRINT NOT (5) does not produce -5. This, also, is tied in with the way numbers are stored in the Sinclair, and will be discussed in a later issue.

By now, it should be obvious that you can't test for zero with IF NOT (N) THEN PRINT "THE NUMBER IS ZERO". If N is 0, NOT (N) will produce a value of -1, making the expression true. In this case, that is what we want. When N is zero, the expression will be true, and the statement "THE NUMBER IS ZERO" will be printed. And if N is -1, there is still no problem, since NOT (-1) will produce 0, making the expression false. But if N is any other number, NOT (N), as we've seen, will return a value other than zero or minus one. And, when evaluating the IF...THEN statement, the Sinclair will consider any value other than zero to be true. So IF NOT (N) THEN PRINT "THE NUMBER IS ZERO", will end up printing the message for any value of N other than minus one.

To sum it up, as long as an expression produces only logical values (0 or -1), you can safely use NOT. Expressions such as IF NOT (A=B) THEN GO TO 10, or IF NOT (X 5 AND Y 8) THEN GO TO 10 are fine. They only deal with logical operators. The value inside the parentheses will be either zero or minus one. But if other integers enter the expression, it's not safe to use NOT. □

Black Hole

Bill Eckel

Black Hole is a game based on a machine language program on my Elf II which uses the RCA 1802 Microprocessor. I understand it was originally called "Teaser." The challenge of rewriting it in Basic for the small memory could not be resisted. The program takes a little over 1K of memory.

Black Hole

You are in space looking at your computer screen which shows a star surrounded by black holes.

```
0 0 0
0 * 0
0 0 0
```

To escape you must get the pattern to be a black hole surrounded by stars.

```
* * *
* 0 *
* * *
```

You can only fire at stars. The stars explode, leaving a black hole, but they also produce new stars in other locations. What the galaxy will look like after you fire at a star is important.

```
1 x .   x 2 x   . x 3   x . .   . x .   . . x   . . . . . .
x x .   . . .   . x x   4 . .   x 5 x   . . 6   x x . . . .
. . .   . . .   . . .   x . .   . x .   . . x   7 x . x 8 x

. . .
. x x
. x 9
```

The number is the star fired upon. The x's are the holes changed to stars or stars changed to holes (the reverse of what they used to be). The .'s are the star or holes that are unaffected.

For convenience sake the computer displays the star numbers in a block next to the galaxy pattern. Examples are:

0 0 0	1 2 3	Shoot star 5	0 * 0	1 2 3
0 * 0	4 5 6	will result in	* 0 *	4 5 6
0 0 0	7 8 9	this	0 * 0	7 8 9

Do not end with all black holes or you will be lost in space forever!

```
0 0 0
0 0 0
0 0 0
```

The lowest possible number of turns to solve the problem is eleven. There are many ways to solve it; here are two:
5.2.8.1.7.3.5.9.2.8.5
5.2.1.3.8.7.5.2.9.8.5
You cannot read the program listing and figure out how to solve it. It is very challenging game which will give hours of fascinating fun.

Bill Eckel, 7336 South 71st Ave., Omaha, NB 68157.

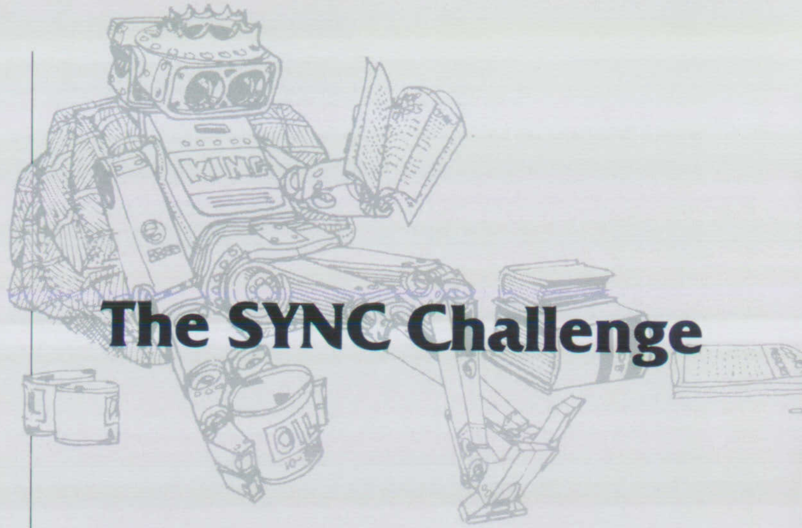
```
5 REM BLACK HOLE WRITTEN BY BILL ECKEL APRIL
  2, 1981
10 DIM X(9)
14 FOR I = 1 TO 9
16 LET X(I) = 0
18 NEXT I
20 LET X(5) = 1
25 CLS
30 PRINT, " BLACK HOLE"
32 PRINT
34 PRINT
35 LET I = 1
40 PRINT ,;
42 FOR A = 1 TO 3
44 IF X(I) = 1 THEN PRINT "*";
46 IF X(I) = 0 THEN PRINT "O";
48 LET I = I + 1
49 PRINT " ";
50 NEXT A
52 PRINT ,; I - 3; " " I - 2, " "; I - 1
53 PRINT
54 IF NOT I = 10 THEN GOTO 40
56 PRINT
58 PRINT
60 GOSUB 1000
65 PRINT "WHICH STAR?"
70 INPUT S
75 IF S   1 OR S   9 THEN GOTO 70
78 IF X(S) = 0 THEN GOTO 70
85 LET X(S) = 0
90 GOSUB S * 100
95 GOTO 25
100 LET B = 2
110 GOSUB 980
120 LET B = 4
130 GOSUB 980
140 LET B = 5
150 GOSUB 980
160 RETURN
200 LET B = 1
210 GOSUB 980
220 LET B = 3
230 GOSUB 980
240 RETURN
300 LET B = 2
310 GOSUB 980
320 LET B = 5
330 GOSUB 980
340 LET B = 6
350 GOSUB 980
360 RETURN
400 LET B = 1
410 GOSUB 980
420 LET B = 7
430 GOSUB 980
440 RETURN
500 LET B = 2
510 GOSUB 980
520 LET B = 4
530 GOSUB 980
540 LET B = 6
550 GOSUB 980
560 LET B = 8
570 GOSUB 980
580 RETURN
600 LET B = 3
610 GOSUB 980
620 LET B = 9
630 GOSUB 980
640 RETURN
700 LET B = 4
710 GOSUB 980
720 LET B = 5
730 GOSUB 980
740 LET B = 8
750 GOSUB 980
760 RETURN
800 LET B = 7
810 GOSUB 980
820 LET B = 9
830 GOSUB 980
840 RETURN
900 LET B = 5
910 GOSUB 980
920 LET B = 6
930 GOSUB 980
940 LET B = 8
950 GOSUB 980
960 RETURN
980 IF X(B) = 0 THEN GOTO 986
983 IF X(B) = 1 THEN X(B) = 0
984 RETURN
986 LET X(B) = 1
990 RETURN
1000 FOR I = 1 TO 9
1010 IF X(I) = 1 THEN GOTO 1050
1020 NEXT I
1025 PRINT "YOU BLEW IT"
1030 PRINT "YOU ARE LOST IN SPACE FOREVER"
1040 STOP
1050 IF X(5) = 1 THEN RETURN
1060 FOR I = 1 TO 4
1070 IF X(I) = 0 THEN RETURN
1080 NEXT I
1090 FOR I = 6 TO 9
1100 IF X(I) = 0 THEN RETURN
1110 NEXT I
1120 PRINT "CONGRATULATIONS"
1130 PRINT "YOU FOUND THE BLACK HOLE"
1140 STOP
```

SAMPLE RUN

```
                BLACK HOLE
          0 0 0        1 2 3
          0 * 0        4 5 6
          0 0 0        7 8 9
WHICH STAR?
5
```

```
                BLACK HOLE
          0 * 0        1 2 3
          * 0 *        4 5 6
          0 * 0        7 8 9
WHICH STAR?
2
```

```
                BLACK HOLE
          * 0 *        1 2 3
          * 0 *        4 5 6
          0 * 0        7 8 9
WHICH STAR?
```



The SYNC Challenge

In our first issue we challenged our readers to fit the Hammurabi game from Creative Computing's *Basic Computer Games* into the 1K memory of the ZX80. A number of readers took us up on the challenge and submitted their entries. The results are as follows:

First place (a one year subscription to *SYNC* and a SYNC T shirt):
Michael Hodgkins
46 Broadway
Duffield
Nr. Derby
DE6 4BU
England

Second place (a one year subscription to *SYNC*):
Un Jung Kang
1620 McElderry St. 12D-4
Baltimore, MD 21205

Third place (a SYNC T shirt):
Ken Berggren
104 Ridgeway Ave.
Louisville, KY 40207

Honorable mentions

Dennis A. Adcock
9516 - 76 Street
Edmonton, Alberta
T6C 2K9
Canada

Lester S. Cottrell, Jr.
108 River Heights Drive
Cocoa, FL 32922

Bob Ferguson
19 Farrington Ave.
Allston, MA 02134

John P. Filley
1501 Murfreesboro Rd.
Nashville, TN 37217

Leonard Gaunt
44 Hartherop Road
Hampton Middlesex TW12 2RP
England

Dennis A. Likens
Box 1125
Tuskegee Institute, AL 36088

Ian S. Logan
24, Nurses Lane
Skellingthorpe
Lincoln LN6 OTT
England

Claude Ostyn
Box 2035
Sitka, AK 99835

Robert M. Selz
P.O. Box 24
Pleasant Plains, IL 62677

Harley Shanko
15025 Vanowen, #209
Van Nuys, CA 91405

Ken Stetina
3626 Eastway Drive
Island Lake, IL 60042

Ephraim Vishniac
38 Gorham Street
Arlington, MA 02174

Hammurabi in 1K

Michael Hodgkins

```
  2 LET Y=1
  4 LET P=100
  6 LET A=1000
  8 LET G=A*3
 10 PRINT "WE HAVE ";G;" BUSHELS"
 15 PRINT "POPULATION ";P
 20 PRINT "CITY OWNS ";A;" ACRES"
 25 LET T=RND(9)+16
 30 IF Y=11 THEN GOTO 250
 35 PRINT "LAND PRICE ";T;" BUSHELS/ACRE"
 40 PRINT "BUY HOW MUCH"
 45 INPUT F
 50 IF F=0 THEN GOTO 70
 55 LET A=A+F
 60 LET G=G-F*T
 65 GOTO 90
 70 PRINT "SELL HOW MUCH"
 75 INPUT F
 80 LET F=-F
 85 GOTO 55
 90 CLS
 95 LET R=RND(9)
100 LET G=G-R*9
105 PRINT "FEED YOUR ";P;" SUBJECTS--MAX ";G
110 INPUT F
112 IF F>G THEN GOTO 110
115 CLS
120 LET G=G-F
125 LET F=P-R-F/20
135 IF F<0 THEN LET F=0
140 PRINT F;" STARVED"
145 IF F>10 THEN GOTO 245
150 LET P=P-F
155 PRINT "SOW GRAIN"
157 PRINT "CITY OWNS ";A;" ACRES"
160 PRINT "WE OWN SEED FOR ";G
165 PRINT P*9 ;" CAN BE TENDED"
170 INPUT F
175 IF F>P*9 OR F>A OR F>G THEN GOTO 170
180 CLS
185 LET G=G+RND(5)*F-F
190 PRINT "YR ";Y
195 IF R=1 THEN GOTO 225
200 PRINT R;" PEOPLE ENTERED CITY"
205 LET P=P+R
210 PRINT "RATS ATE ";R*9;" BUSHELS"
215 LET Y=Y+1
220 GOTO 10
225 PRINT "PLAGUE"
230 PRINT P-P/2;" DIE"
235 LET P=P/2
240 GOTO 210
245 PRINT "MURDERER"
250 PRINT "END OF REIGN"
```

The game begins by telling the player the state of the economy he is to manage. The lines of the game function as follows:

2-8	Set up variables, year, population, acreage, and grain.
10-20	PRINT information on the screen.
25	Sets random land price.
30	Year counter to check for end of reign.
35-45	PRINTs land price and asks how much the player wishes to buy.
50	If no land is bought, the program jumps to 70.
55-60	Adjusts acres and grain according to the transaction.
70-85	Asks how much land the player wants to sell; then goes to 55 to adjust variables as before.
95-100	Chooses a random number to be used for a variety of things. First it subtracts the amount of grain eaten by rats.
105-145	Tells the player to feed subjects; checks that he is not feeding them more grain than he has; works out how many people starved and the remaining grain. If the number starved is too large, it jumps to 245.
155-170	Instructions to sow grain, telling player how much grain may be sown, tended, etc.
175	Checks that he is not cheating.
185	Decides crop to be harvested.
190-210	Gives readout of year's events.
215	Increments year number by 1.
220	Returns to beginning of new year.
225-240	Plague routine; approximately half the people die.
245	Prints MURDERER if more than 10 people have starved.
250	Final statement; game over.

To run, press RUN and NEWLINE. Then enter your choices as called for by the computer.

When you reach "END OF REIGN" and want to start again, press any key and then press RUN and NEWLINE.

The program uses almost every byte of memory available. The same variable F is used for all INPUT's to save memory and also a single random number R is used for the number of people entering the city, the number of bushels devoured by rats, and the test for a plague. A plague occurs when R=1, thus avoiding the complications caused by "1 PEOPLE" entering the city. □

How Is It Done? *A series of articles*

Auto-Display-Changing

Dr. I. S. Logan

Introduction

The standard ZX80 is supplied with a 4K ROM and 1K of memory. There are 22 commands that can be used in the Basic provided in the 4K ROM. However, there is no command that will cause the display to be shown for a specified length of time before a change is made to show the next display. The advertisements for the 8K ROM mention a command called PAUSE which is expected to perform this function.

The following program constructs the machine code routine for such a PAUSE command. The actual program occupies about 1/4K, leaving the programmer a little under 3/4K in which he can store his different displays.

The actual displays are constructed using Basic PRINT commands in the version given here, but there is no reason why machine code constructed displays should not be used if greater speed or complexity is required. (See *SYNC*, vol. 1, no. 1)

A certain amount of 'flicker' is produced between displays because the routine 'returns to Basic' after the specified time period. This 'flicker' can be eliminated only by remaining in machine code and synchronizing the program perfectly.

The Theory

The Screen and Keyboard routine, decimal address 316-437, Hex. address 013C-01B5, in the 4K monitor program can be considered the dominant routine in the operation of the ZX80.

It is this routine that both reads the Keyboard and produces the display on the TV screen. It therefore follows that this routine cannot be called unless a complete display file has already been constructed.

The routine can be divided into three parts:

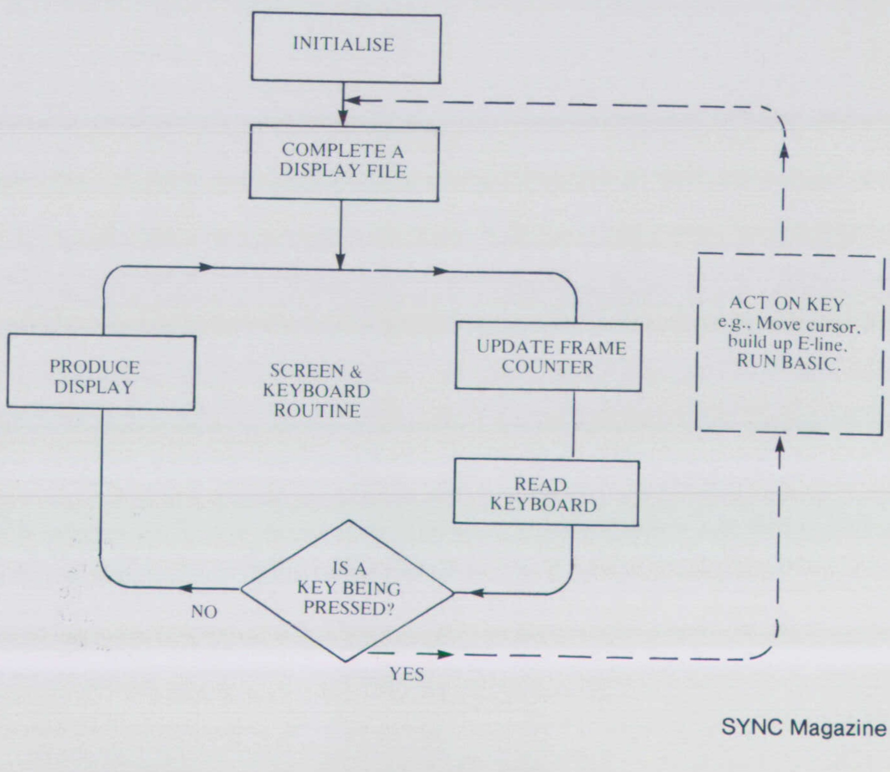
Part 1. Update the frame counter.

Part 2. Test the Keyboard for new input.

Part 3. Produce the display of the current display file.

If there is no key being pressed, then the whole routine is executed over and over again. However, if a key is being pressed, then an exit is made from the routine to handle the 'interruption.' This may lead to the cursor being moved, characters being added to the current E-line, or the RUNning of a Basic program. As long as the programmer has not created a 'never ending loop,' the Screen and Keyboard routine will eventually be re-entered and a display will again appear on the TV screen.

The flow diagram below illustrates the normal operation of the ZX80.

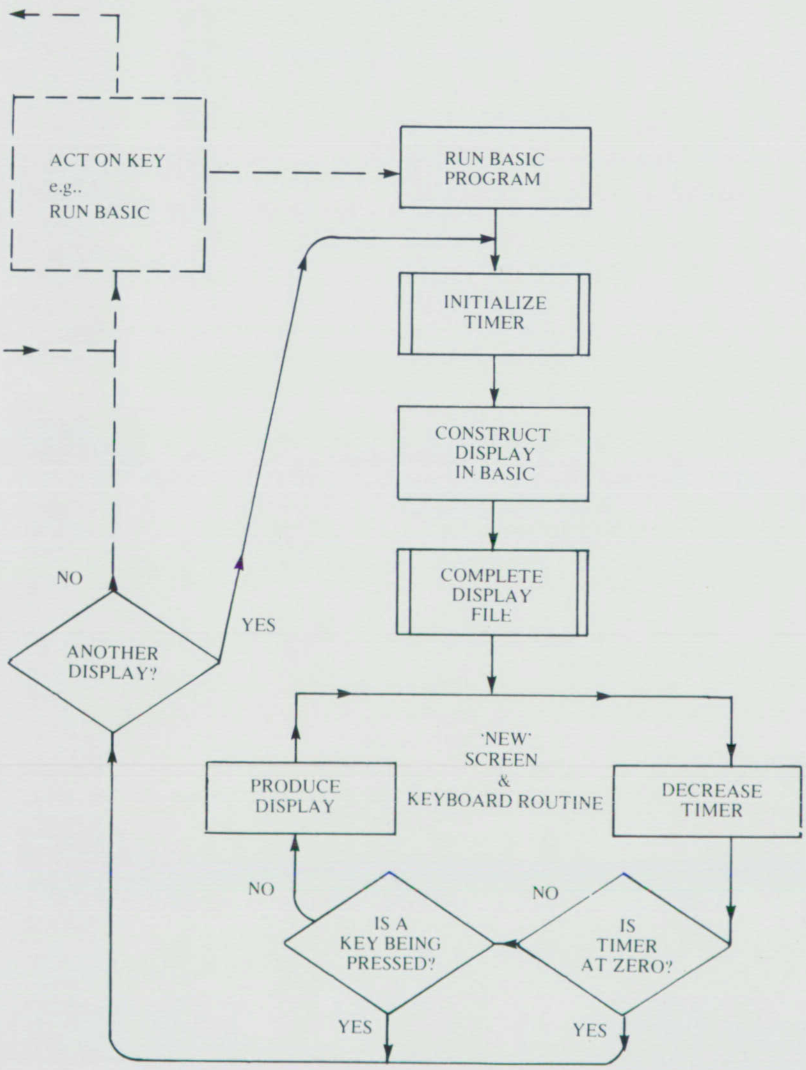


Dr. I. S. Logan, 24 Nurses Lane, Skellingthorpe, Lincoln LN6 OTT England. This article is the second in a series.

The AUTO-DISPLAY-CHANGING program copies most of the Screen and Keyboard routine from the 4K monitor program into the memory and adds a timing loop so that the display can be held for up to 256 frames, about 5 seconds. The programmer is then able to use this 'new' routine to produce a display on the TV screen.

It is important to emphasize again that a complete display file of 24 lines must be constructed before the routine is called. In the following program the display file is completed by calling the subroutine at line 16. However, this can also be done in machine code if required.

The following flow diagram shows how the ZX80 operates with the 'new' routine:



The Program

The program, although fairly simple, is quite difficult to enter. Therefore do it slowly and carefully. SAVE the partly entered program *often*.

Step 1

Enter the following lines and then SAVE:

```
  2 REM 123456789012345678901234567890123456789
0123456789012345678901234567890123456789
0123456789012345678901234567890123456789
0123456789012345678901234567890123456789
0123456789012345678901234
  4 GO TO 28
  6 LET A=USR(16428)
  8 CLS
```

Lines 2–4: Reserve 134 locations. Do not use all spaces!

Line 6: A will return the Keyboard codes.

From now on NEVER use LIST or HOME.

Step 2

Make the following check:
Enter as a direct command:
PRINT PEEK(64*256+199) & NEWLINE and the value 118 should appear. (This is the 'end of line marker' for line 6 and the correct address must be known.)

Step 3

Enter the following lines and then SAVE.

10 LET A=16428	
12 POKE A,205 14 POKE A+1,59 16 POKE A+2,64	Call the routine at line 42
18 POKE A+3,33 20 POKE A+4,199 22 POKE A+5,64	The address of the end of line 6
24 POKE A+6,34 26 POKE A+7,38 28 POKE A+8,64	The return address needs to be stored in System Variable 16422 and 16423
30 POKE A+9,96 32 POKE A+10,105	Return Keyboard codes in HL register pair.
34 POKE A+11,201	
36 POKE A+12,205 38 POKE A+13,173 40 POKE A+14,1	Call screen production subroutine.
42 POKE A+15,58 44 POKE A+16,43 46 POKE A+17,64	Fetch timer from 16427.
48 POKE A+18,61	Decrement timer.
50 POKE A+19,200	Exit if timer is zero.
52 POKE A+20,50 54 POKE A+21,43 56 POKE A+22,64	Restore timer.
58 FOR I=319 TO 427 60 POKE 16132+I,PEEK(I) 62 NEXT I	Copy most of Screen and Keyboard routine from monitor to the memory.
64 POKE A+24,4	Adjust timing slightly.
66 POKE A+132,135	Change a JR value.
68 LET T=0 70 FOR I=A TO A+132 72 PRINT PEEK(I); 74 LET T=T+PEEK(I) 76 NEXT I 78 PRINT 80 PRINT "CHECKSUM = ";T SAVE.	Form a CHECKSUM

Now enter RUN 10 & NEWLINE. The screen will now display the machine code that has been entered into line 2 and held off the screen.
The correct CHECKSUM is 14421; correct any errors before proceeding.

Step 4

Delete all the lines from 10 to 80 (inclusive) by entering the line number and NEWLINE over and over again.

SAVE the program. It should consist of lines 4-8 on the screen and line 2, off the screen.

Step 5

Enter the rest of the Basic program:

10 POKE 16427,255	Initialize timer to 5 seconds.
12 POKE 16421,24	Create a 24th line so as to give a 'full display.'
14 RETURN	
16 LET A=PEEK(16421)-1 18 IF A=0 THEN RETURN 20 FOR A=1 TO A 22 PRINT 24 NEXT A	This routine will complete the display file. It adds the appropriate number of PRINT's to fill the 24 lines.
26 RETURN 28 GO SUB 8	Initialize timer for 1st display.
30 REM AUTO-DISPLAY-CHANGING	Optional REM line.

The program is now complete, so SAVE this version carefully. Remember, never use LIST or HOME.

Using the program

It is not really the author's intention in this article to describe at any great length just how the program can be used. The following examples are given so that the reader can start to see for himself how different problems are tackled.

Simple display changing

Enter the lines;

```
100 PRINT "DISPLAY ONE"       } A very simple 1 line display.
196 GO SUB 16                 } Complete display file.
198 GO SUB 6                  } Produce the display.

200 PRINT "DISPLAY TWO"       } Another simple display.
296 GO SUB 16                 } Complete the 2nd display.
298 GO SUB 6                  } Produce the display.
300 GO TO 100                 } LOOP BACK
996 CLS
998 STOP
RUN
```

As long as the program is entered correctly, the first display should appear on the screen for five seconds. Then the screen will 'flicker' and the second display will appear. Because of the LOOP BACK the displays will alternate forever!

Note that all the keys are active. Pressing any key, except BREAK, will cause a switch to the next display. The BREAK key is still active as it is tested at the end of each Basic line. This key can therefore be used to 'exit' from the LOOP.

The largest possible display

The following lines show that there are about 550 locations still available for the displays in the standard 1K ZX80.

```
100 FOR I=1 TO 550            Display is to have 550 '*'s.
102 PRINT "*";
104 NEXT I
106 PRINT                     Always go to the next line.
196 GO SUB 16                 Complete display.
198 GO SUB 6                  Produce display.
200 GO TO 198                 LOOP BACK.
996 CLS
998 STOP
```

In the above program the LOOP BACK is used in a different way. By repeating line 198 over and over again, the current display file is used again without any changes.

Find the number

The following game shows how the A variable returns the keyboard code.

In the program a random number in the range 1-5 is the first generated. Then the keys that are pressed by the player are tested for the correct key value.

Each time a key is pressed the score is incremented. At the end of each 5 second period without a key stroke the score is also incremented.

```
100 REM FIND THE NUMBER
102 LET N=RND(5)
104 LET N=521*(N=1) OR 1033*(N=
   2) OR 2057*(N=3) OR 4105*(N=4) O
   R 8201*(N=5)
106 LET T=1
108 GO TO 204
200 PRINT "SORRY. TRY AGAIN"
202 LET T=T+1
204 PRINT
206 PRINT "I KNOW THE KEY. DO
   YOU? (1-5)"
296 GO SUB 16
298 GO SUB 6
300 IF NOT A=N THEN GO TO 200
302 PRINT "WELL DONE"
304 PRINT "YOU TOOK";T;" GO";
306 IF NOT T=1 THEN PRINT "ES"
308 PRINT
310 PRINT "PRESS NEWLINE TO RES
   TART"
312 INPUT A$
314 IF NOT A$="" THEN STOP
316 CLS
318 RUN
```

The Key values

The Screen and Keyboard routine scans the keyboard and returns in the BC register pair a KEY VALUE that is different for every stroke. As there are 78 keystrokes, there are 78 different key values.

In the 'Build up an E-line' routine these key values are changed to the range 1 to 78; then the look up table is used to find the correct ZX80 character codes.

However, in order to make the AUTO-DISPLAY-CHANGING program as short as possible, the conversion of key values to character codes has not been included.

The following program can be used to show the KEY VALUES:

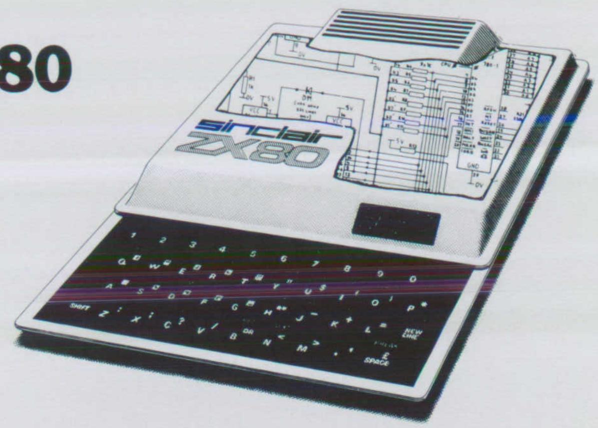
```
100 REM KEY VALUES
102 LET A=0
104 PRINT A
196 GO SUB 16
198 GO SUB 6
200 GO TO 104
```

Conclusion

Many other kinds of programs can be written using the AUTO-DISPLAY-CHANGING routine. The author has a very nice digital clock, but the 'flicker' is a little annoying. Much of the background work for this article is discussed in the author's *The ZX80 Companion* which contains a more elementary version of this particular program.

Looking inside the ZX80

by Harley Shanko



Since the materials supplied with the ZX80 had no machine language examples, I decided to write routines to let Basic show me the ROM contents. These routines resulted from that effort; later they were combined to permit switching from one mode to another.

The object code routine OBJ allowed me to generate a 'hand-disassembled' listing of the 4K ROM, and SYMB to see the 'printables'—this allows locating the Basic statement look-up table, single key codes expansion, and the 'integral function' expressions. CODE allows a look into the details of how the Basic lines are stored (note: constants are stored in decimal form as entered, unlike some Basic's) and permits easy counting of the number of bytes consumed by each line.

Use of the program is straightforward. After RUN, the selection is displayed. The operation and keyboard activity are as follows:

Selection:	Enter number (1 to 3) of desired listing (plus NEWLINE); the address is then requested—enter decimal value of address beginning.
Continue:	Hit NEWLINE.
Change:	To change selection, hit any other key (except SPACE or NEWLINE) plus NEWLINE to return to selection mode.
Exit:	After listing and cursor returns, hit SPACE then NEWLINE twice.

The display is in standard format, with the address in hex at the left and data contents to the right. OBJ presents a cluttered display. Since the program uses about 1/2 K and the displayed information uses 336 bytes, formatting with spaces between bytes (hex-pairs) can only be done by either displaying fewer lines or using OBJ as a stand-alone program to increase readability. SYMB is better as it is textual. Look at ROM beginning at 006C (108 decimal) for the keyboard matrix decoder, 00BD (189) for the single-key code expansion, 0BC0 (3008) for the integral function decoder, or at RAM at 4028 (16424) to see this program as stored by the ZX80. Use of CODE at 4028+ details exactly each byte of the program.

Harley Shanko, 15025 Vanowen, #209, Van Nuys, CA 91405.

```
zx80--looking inside the zx80

                ZX80 (1K) MEMORY LISTER.          Selection

 10 PRINT "1=OBJ 2=SYMB 3=CODE"
 15 REM BY H SHANKO-22FEB81
 20 INPUT S
 22 CLS
 24 IF S>3 THEN GO TO 10
 26 PRINT "ADDR="
 28 INPUT N                                       Test if selection OK
 30 LET P=S*100                                   Enter beginning address now
 52 CLS
 54 FOR A=1 TO 16                                 Set selection GOSUB address
 56 IF S=3 THEN FOR A=1 TO 20
 60 LET X=N                                       Set #lines to 16
 62 GO SUB 80                                               or 20 lines
 64 PRINT " ";                                    Get address
 66 GO SUB P                                          and display it
 68 PRINT                                             plus SPACE
 70 NEXT A                                        Go to selection subroutine
 72 INPUT N$                                         do "CRLF" for each line
 74 IF N$="" THEN GO TO 52                           until done
 75 IF N$=" " THEN STOP                           Get keyboard entry
 76 GO TO 10                                          test if "continue"
                                                      or "stop"
 80 PRINT CHR$(X/4096+28)                         Otherwise is "new selection"
 82 LET X=X-(X/4096)*4096
 84 PRINT CHR$(X/256+28);                         4-hex entry for decimal-hex conv.
 86 LET X=X-(X/256)*256                              calculate next hex digit
 88 PRINT CHR$(X/16+28);                          3-hex entry
 90 LET X=X-(X/16)*16
 92 PRINT CHR$(X+28);                             2-hex entry
 94 RETURN
 95 LET X=PEEK(N)                                 1-hex entry
 97 GO TO 88
                                                  Get byte at address
100 FOR L=1 TO 8
110 GO SUB 95                                     Set for 8 bytes/line
120 GO TO 250                                        get byte
                                                     and loop until done
200 FOR L=1 TO 16
210 LET X=PEEK(N)                                 Set for 16 char/line
220 IF X>127 THEN LET X=X-128                     Get byte
230 IF X<12 OR X>63 THEN LET X=0                     remove if "inverted video"
240 PRINT CHR$(X)                                    exclude non-textual char.
250 LET N=N+1                                     Display char.
260 NEXT L                                        Increment address
270 RETURN                                           and loop until done

300 GO SUB 95

310 PRINT,CHR$(PEEK(N));                          Get byte at address, display it
320 LET N=N+1                                        do SPACE and display CODE
330 RETURN                                        Get next address
```

A byte-search routine BYT was used before disassmebling the ROM to locate data, such as Z80 unconditional CALL, JumP, RETurn addresses, although it will search for any byte. Unfortunately, adding BYT, like most other routines to the program, will cause OBJ and SYMB selections to bomb-out with "4" type errors, not enough memory, before completing the display. However, CODE and BYT will still be usable. An alternate solution is to make a second file by deleting a selection and its routine and substituting BYT. To add BYT, follow printout labeled Figure 1.

Another routine generated was a ZX80 dot-matrix routine DOT to study the characters at an 8x scale; the characters are 7 x 6 in an 8 x 8 box and DOT forms a SPACE or inverted one for each dot in the character. As a fifth selection, modify the program so that printout will appear as in Figure 2.

Figure 1.

```
change   10    add ...4=BYT... to print statement
         24    change ...S>3... to ...S>4...
         56    change ...S=3... to ... S>2...
add      36 IF S<4 THEN GO TO 52
         38 PRINT "BYTE=";
         40 INPUT B
         42 GO SUB 410
        400 LET N=N+1                 Incr. address
        410 IF NOT PEEK(N)=B THEN GO TO 400  Loop until match
        420 RETURN
```

Figure 2.

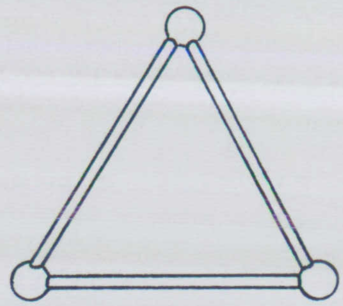
```
change   10  add ...5=BYT...
         24  change ...S>3 (or 4)... to S>5...
add     500 LET X=PEEK(N)            Get byte
        510 FOR L=1 TO 8             Do 8 dots/line
        520 LET C=27                 "space"
        530 IF X<128 THEN GO TO 560  Test ms bit
        540 LET C=128                If ONE, invert
                                           "space"
        550 LET X=X-128              Remove msb, if
                                           ONE
        560 LET X=X*2                Do left shift
        570 PRINT CHR$(C);           Display dot/no dot
        580 NEXT L                   Loop until done
        585 LET N=N+1                Then incr. address
        590 RETURN
```

Line 520 displays a 'period' for the character dot location where the dot should be OFF; this provides a reference for the character location in the box. For a true representation, change 520 to LET C=0. If a 0,1 type (binary) display is desired, change C=28 at 520 and C=29 at 540.

The ZX80 dot matrix is located at 0E00 through 0FFF; thus to see the characters enter decimal 3584 (=0E00). Because of their size, only two characters per display are possible. □

puzzles & problems

A Building Problem



Our first problem today is an interesting test in construction. In the illustration at the right we see a triangle that has been constructed using three matchsticks and three balls of clay. In our puzzle you are given nine matchsticks and as much clay as you need to connect them together in such a manner as to form seven equilateral triangles. You are not allowed to cross or break the matchsticks. Merlin will be by shortly to inspect your construction.

The Lucky Number

Many persons have what they consider a "lucky" number. Show such a person the row of figures subjoined — 1, 2, 3, 4, 5, 6, 7, 9 (consisting of the numerals from 1 to 9 inclusively, with the 8 only omitted) — and inquire what is his lucky or favorite number. He names any number he pleases from 1 to 9, say 7. You reply that, as he is fond of sevens, he shall have plenty of them, and accordingly proceed to multiply the series given above by such a number that the resulting product consists of sevens only.

Required, to find for each number that may be selected the multiplier which will produce the above result. (From *Merlin's Puzzler)*

The Puffer-Belly Problem

A Conrail passenger and freight train out of Hoboken, New Jersey, was heading west towards Morristown, home of Creative Computing, at a speed of 45 miles per hour. Along the way the train meets and is passed by a Dover local train heading east at 36 miles per hour. An alert passenger on the Conrail train, for some reason unknown to us, clocks the Dover train as it passes by him. He finds that it takes exactly 6 seconds for the Dover train to pass by his window. Using the information above, can you calculate the length of the Dover train?

A Seven-Letter Charade

Alright, Hermione, let's see what you learned this year at St. Trinian's. From the four following hints you should be able to deduce the word asked for by this charade. The numbers refer to positional letters within the word."

"My 1, 2, 7 is an extreme point."
"My 3, 4, 5, 7 is what the reader will be when he solves this puzzle."
"My 5, 2, 3, 1, 4 is in heaven."
"My 4, 5, 6, 7 is the earth."
"My whole is a country in Europe."

The Three Jealous Husbands

Three jealous husbands traveling with their wives find it necessary to cross a stream in a boat which holds only two persons. Each of the husbands has a great objection to his wife crossing with either of the other male members of the party unless he himself is also present.

How is the passage to be arranged?

The Four Jealous Husbands

Arithmeticians have racked their brains to devise a means of transit for four husbands and four wives under the same conditions stated in the previous puzzle, but, with a boat holding two persons only, the problem is insoluble. If we suppose, however, that the boat contains three persons, it may be solved.

How is this passage to be arranged?

(The two puzzles above are from *Merlin's Puzzler 3*)

If you have a favorite puzzle that you would like to share with our readers, then send it along to Merlin. If he uses your puzzle, he will send you a copy of one of his famous puzzle and game books. If you can't wait that long, then you can buy these books from Creative Computing. Each book, *Merlin's Puzzler, Merlin's Puzzler 2,* and *Merlin's Puzzler 3,* is 128 pages long and a big 8½" x 11" in size. There's a world of "puzzling" entertainment in these three volumes.

Until next time, Merlin says, "Have a happy   !" You fill in the blank.

Your editor, Charles Barry Townsend

Answers on page 35.

The story behind the two best selling computer games books in the world.

Computer Games

by David H. Ahl

Everybody likes games. Children like tic tac toe. Gamblers like blackjack. Trekkies like Star Trek. Almost everyone has a favorite game or two.

It Started in 1971

Ten years ago when I was at Digital Equipment Corp. (DEC), we wanted a painless way to show reluctant educators that computers weren't scary or difficult to use. Games and simulations seemed like a good method.

So I put out a call to all our customers to send us their best computer games. The response was overwhelming. I got 21 versions of blackjack, 15 of nim and 12 of battleship.

From this enormous outpouring I selected the 90 best games and added 11 that I had written myself for a total of 101. I edited these into a book called 101 Basic Computer Games which was published by DEC. It still is.

When I left DEC in 1974 I asked for the rights to print the book independently. They agreed as long as the name was changed.

Contents of *Basic Computer Games* (right) and *More Basic Computer Games* (below).

Introduction
The Basic Language
Conversion to Other Basics
Acey Ducey
Amazing
Animal
Awari
Bagels
Banner
Basketball
Batnum
Battle
Blackjack
Bombardment
Bombs Away
Bounce
Bowling
Boxing
Bug
Bullfight
Bullseye
Bunny
Buzzword
Calendar
Change
Checkers
Chemist
Chief
Chomp
Civil War
Combat
Craps
Cube
Depth Charge
Diamond
Dice
Digits
Even Wins
Flip Flop
Football
Fur Trader
Golf
Gomoko
Guess
Gunner
Hammurabi
Hangman
Hello
Hexapawn

Hi-Lo
High I-Q
Hockey
Horserace
Hurkle
Kinema
King
Letter
Life
Life For Two
Literature Quiz
Love
Lunar LEM Rocket
Master Mind
Math Dice
Mugwump
Name
Nicomachus
Nim
Number
One Check
Orbit
Pizza
Poetry
Poker
Queen
Reverse
Rock, Scissors, Paper
Roulette
Russian Roulette
Salvo
Sine Wave
Slalom
Slots
Splat
Stars
Stock Market
Super Star Trek
Synonym
Target
3-D Plot
3-D Tic-Tac-Toe
Tic Tac toe
Tower
Train
Trap
23 Matches
War
Weekday
Word

Artillery-3
Baccarat
Bible Quiz
Big 6
Binary
Blackbox
Bobstones
Bocce
Boga II
Bumbrun
Bridge-It
Camel
Chase
Chuck-A-Luck
Close Encounters
Column
Concentration
Condot
Convoy
Corral
Countdown
Cup
Dealer's Choice
Deepspace
Defuse
Dodgem
Doors
Drag
Dr. Z
Eliza
Father
Flip
Four In A Row
Geowar
Grand Prix
Guess-It
ICBM
Inkblot
Joust
Jumping Balls
Keno
L Game

Life Expectancy
Lissajous
Magic Square
Man-Eating Rabbit
Maneuvers
Mastermind
Masterbagels
Matpuzzle
Maze
Millionaire
Minotaur
Motorcycle Jump
Nomad
Not One
Obstacle
Octrix
Pasart
Pasart 2
Pinball
Rabbit Chase
Roadrace
Rotate
Safe
Scales
Schmoo
Seabattle
Seawar
Shoot
Smash
Strike 9
Tennis
Tickertape
TV Plot
Twonky
Two-to-Ten
UFO
Under & Over
Van Gam
Warfish
Word Search Puzzle
Wumpus 1
Wumpus 2

Converted to Microsoft Basic

The games in the original book were in many different dialects of Basic. So Steve North and I converted all the games to standard Microsoft Basic, expanded the descriptions and published the book under the new name Basic Computer Games.

Over the next three years, people sent in improved versions of many of the games along with scores of new ones. So in 1979, we totally revised and corrected Basic Computer Games and published a completely new companion volume of 84 additional games called More Basic Computer Games. This edition is available in both Microsoft Basic and TRS-80 Basic for owners of the TRS-80 computer.

Today Basic Computer Games is in its fifth printing and More Basic Computer Games is in its second. Combined sales are over one half million copies making them the best selling pair of books in recreational computing by a wide margin. There are many imitators, but all offer a fraction of the number of games and cost far more.

The games in these books include classic board games like checkers. They include challenging simulation games like Camel (get across the desert on your camel) and Super Star Trek. There are number games like Guess My Number, Stars and Battle of Numbers. You'll find gambling games like blackjack, keno, and poker. All told there are 185 different games in these two books.

Whether you're just getting started with computers or a proficient programmer, you'll find something of interest. You'll find 15-line games and 400-line games and everything in between.

The value offered by these books is outstanding. Every other publisher has raised the price of their books yet these sell for the same price as they did in 1974.

Moneyback Guarantee

Examine one or both of these books and key some games into your computer. If you're not completely satisfied we'll refund the full purchase price plus your return postage.

Basic Computer Games costs only $7.50 and More Basic Computer Games just $7.95 for either the Microsoft or TRS-80 edition (please specify your choice on your order). Both books together are $15. Send payment plus $2.00 shipping and handling to Creative Computing Press, Morris Plains, NJ 07950. Visa, MasterCard and American Express orders should include card number and expiration date. Charge card orders may also be called in toll-free to 800-631-8112 (in NJ 201-540-0445).

Order today to turn your computer into the best game player on the block.

creative computing

Morris Plains, NJ 07950
Toll-free **800-631-8112**
(In NJ 201-540-0445)

Mastermind 2K

Raymond Fowkes

This game is played much like the original board version. The computer selects a code of four colors from a possible six: red, orange, yellow, green, blue, and white (duplicates allowed). It is now up to you to find, in nine moves or less, the exact color and position of each element of the code by entering four colors of your choice, first letters only, one at a time. The computer then compares the guess with the pre-selected code, first for black pegs and then for white pegs. The pegs are displayed next to the corresponding guess. A black peg means a right color in the right position, a white peg means a right color in the wrong position.

For example, suppose the hidden code was R B W B, and the first guess was R G B O. You would be given one black peg and one white peg because 'red' is in the right position and 'blue' is correct but in the wrong position.

These features make this program superior to many other versions of this game.

The game continues until 1) the code is broken and "uncovered," 2) "Q" is entered, signifying 'quit,' or 3) all nine tries are used.

Conversion

This program is designed for a 2K system, but can be reduced to 1K by doing the following:

Delete lines 10-140, 250-320, 720-770.

Change lines:

```
330 PRINT "ROYGBW   " (3 sp.)
480 POKE C+214+2*E,B(E)
560 POKE C+203+E,9
660 POKE C+203+E,128
800 POKE C+D*20+2*B+11,0
810 POKE C+D*20+2*B+12,A(B)
830 POKE C+D*20+19,0
840 STOP
```

Raymond Fowkes, P.O. Box 336, Coalinga, CA 93210.

```
  5 RANDOMIZE
 10 PRINT ,"  MASTERMIND" (After " leave 2 sp.)
 20 PRINT
 30 PRINT "  I WILL SELECT A CODE OF FOUR" (2 sp.)
 40 PRINT "COLORS AND YOU MUST TRY TO BREAK IT USING THE CLUES I GIVE YOU."
 50 PRINT
 60 PRINT "   A BLACK PEG MEANS A RIGHT" (3 sp.)
 70 PRINT " COLOR IN THE RIGHT POSITION." (1 sp.)
 75 PRINT
 80 PRINT "  A WHITE PEG MEANS A RIGHT" (2 sp.)
 85 PRINT " COLOR IN THE WRONG POSITION." (1 sp.)
 90 PRINT
100 PRINT "      YOU HAVE 9 TRIES." (6 sp.)
110 PRINT
115 PRINT "    INPUT Q TO QUIT AND/OR","       P TO PLAY AGAIN." (4 sp.and 7 sp.)
120 PRINT
125 PRINT "      TYPE IN  P  TO PLAY." (6 sp.)
130 INPUT A$
135 IF NOT A$="P" THEN STOP
140 CLEAR
150 DIM A(3)
160 DIM B(3)
170 DIM C(3)
180 FOR B=0 TO 3
190 LET A(B)=RND(6)
200 IF A(B)=1 OR A(B)=2 OR A(B)=3 THEN LET A(B)=A(B)*8+36
210 IF A(B)=4 THEN LET A(B)=39
220 IF A(B)=5 OR A(B)=6 THEN LET A(B)=A(B)*7+20
230 NEXT B                                   (Generate code)
250 CLS
300 PRINT "B = BLUE  Y = YELLOW  O = ORANGER = RED   G = GREEN  W = WHITE
310 PRINT " ■ = BLACK PEG",CHR$(128);" = WHITE PEG
320 PRINT
330 PRINT ," ";              ■ Shift A)
340 FOR F=1 TO 11
360 PRINT CHR$(128);
370 NEXT F
380 FOR C=1 TO 9
390 PRINT
400 PRINT
410 PRINT 10-C,,"  ";
420 NEXT C
430 FOR D=1 TO 9
440 FOR E=0 TO 3
450 INPUT A$
460 LET B(E)=CODE(A$)
470 GO SUB 900
480 POKE C+310+2*E,B(E)
490 IF A$="Q" THEN GO TO 720
500 LET C(E)=A(E)
510 NEXT E
520 LET E=0
530 FOR B=0 TO 3
540 IF NOT B(B)=C(B) THEN GO TO 590 (Check for black peg)
550 LET E=E+2
560 POKE C+299+E,9
570 LET B(B)=0 (Destroy matching pairs)
580 LET C(B)=1
590 NEXT B
600 IF E=8 THEN GO TO 780
610 FOR B=0 TO 3
620 FOR F=0 TO 3
630 IF C(B)<2 THEN GO TO 700
640 IF NOT C(B)=B(F) THEN GO TO 690
650 LET E=E+2          (Check for white pegs)
660 POKE C+299+E,128
670 LET B(F)=0
680 GO TO 700
690 NEXT F
700 NEXT B
710 NEXT D
720 LET A$=" TOO BAD"
730 FOR B=97 TO 103
740 LET A$=TL$(A$)
750 GO SUB 900
760 POKE B+C+D*20,CODE(A$)
770 NEXT B
780 GO SUB 900
790 FOR B=0 TO 3
800 POKE C+D*20+2*B+107,0
810 POKE C+D*20+2*B+108,A(B)
820 NEXT B
830 POKE C+D*20+115,0
840 GO TO 130
900 LET C=PEEK(16396)+PEEK(16397)*256-D*20
910 RETURN
```

[Instructions]

Sample Runs (2K)

MASTERMIND

I WILL SELECT A CODE OF FOUR COLORS AND YOU MUST TRY TO BREAK IT USING THE CLUES I GIVE YOU.

A BLACK PEG MEANS A RIGHT COLOR IN THE RIGHT POSITION.

A WHITE PEG MEANS A RIGHT COLOR IN THE WRONG POSITION.

YOU HAVE 9 TRIES.

INPUT Q TO QUIT AND/OR P TO PLAY AGAIN.

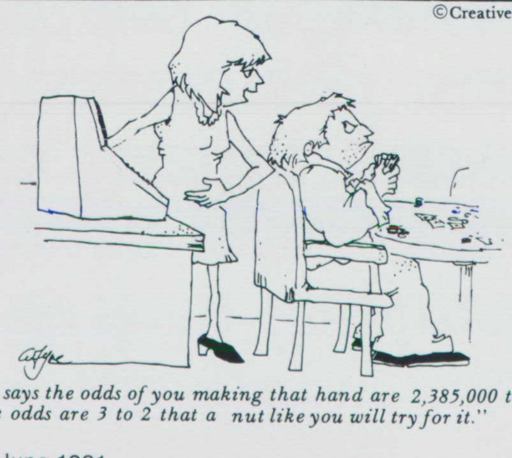
TYPE IN P TO PLAY.

B = BLUE Y = YELLOW O = ORANGE
R = RED G = GREEN W = WHITE
■ = BLACK PEG □ = WHITE PEG

Y R B W

[Code is only "uncovered" after it is guessed, all turns are used, or a quit]

```
9
8
7
6 ■ ■ ■ ■ Y R B W    [Correct answer]
5 ■ □     Y Y G R
4 ■ ■     Y Y O W
3 ■       G W G W    ■ is chr 9
2 ■       O R O R    □ is chr 128
1 □ □     B B Y Y
"P"
```

"It says the odds of you making that hand are 2,385,000 to 1, and the odds are 3 to 2 that a nut like you will try for it."

B = BLUE Y = YELLOW O = ORANGE
R = RED G = GREEN W = WHITE
■ = BLACK PEG □ = WHITE PEG

TOO BAD □ R B O G □

```
9
8
7         Q          [Quit]
6 ■ □ □ □ B G O R
5 □ □ □ □ B R G O
4 ■ □ □ □ B O R G
3 ■ ■     R R R G
2 ■ □     O O O B
1 ■       B B B Y
9:135
```

Sample Runs (1K)

ROYGBW □ G R G Y □

[Code is only "uncovered" after it is guessed, a quit, or all turns are used]

```
9
8
7
6 ■ ■ ■ ■ G R G Y    [Correct answer]
5 ■ □     G W B R
4 ■ ■     G W O Y
3 ■       Y B B Y
2 □       R O R O
1 ■ □     G G W W
9:840
```

Graphics Surprises

James H. Parsons

When we combine the uncertainty of the ZX80's randomizing feature with its graphics capacities, we have the ingredients for a lot of fun. For example, the Crazy Quilt program uses only two instructions, yet it fills the screen with a zany tangle of symbols and spots:

Crazy Quilt

```
10 PRINT CHR$(RND(9))+2);
20 GO TO 10
```

Using "((RND(9)+2)" to assign the characters to be printed insures that only the keyboard graphics symbols, CHR$(2) through CHR$(11), will be selected; and it avoids the blank space, CHR$(0), and the null string, CHR$(1). By changing the range of numbers being randomly selected, we can fill the screen with letters, numbers, punctuation marks, inverse characters, or any combination of these so try numbers other than 9 and/or 2.

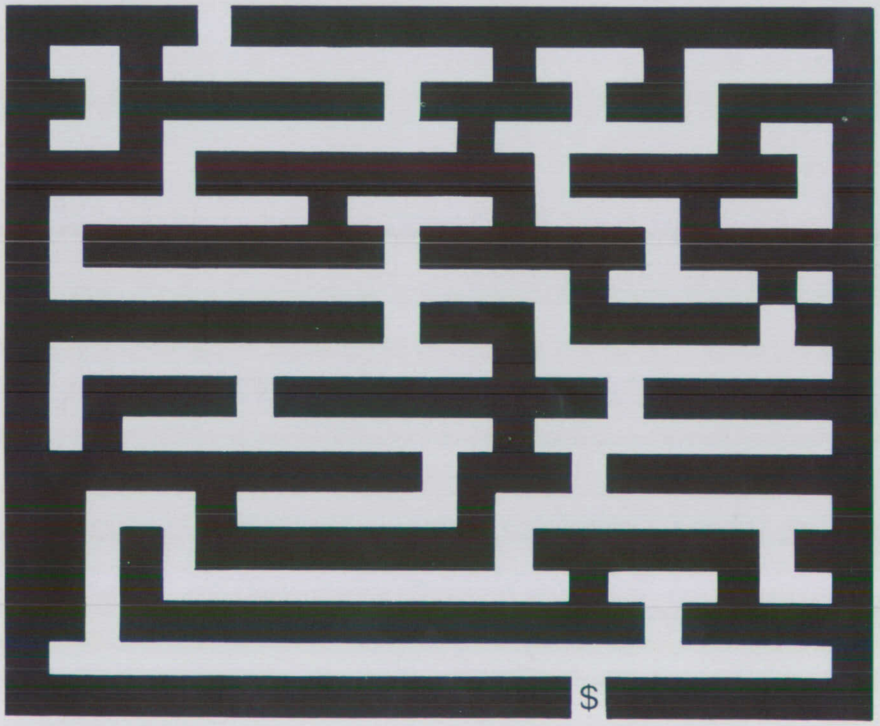
Walls and Dikes

A more challenging application of graphics and randomizing is found in "Walls and Dikes." This program generates a maze in which the configuration of the baffles is fixed randomly within parameters which the player can set. In order to make spaces for traveling within the maze, the program alternates rows of "walls" with rows of "dikes." The wall rows are solid with just a few randomly-placed spaces to pass through. The dike rows are completely open, except for several randomly-placed dikes. To keep maze travelers from sneaking around the ends of wall lines, there is a 19 x 23-space frame around the whole maze. A randomly-placed opening in the top of the frame lets maze travelers in. In the bottom of the frame is a randomly-located treasure marked by a $.

The prospective maze traveler enters the densities (from 1 to 100) of the walls and dikes. When the computer draws a maze according to the densities specified, the maze traveler must try to find a way from the door at the top of the frame through the maze to the treasure at the bottom. Relatively "thin" densities of walls and dikes (for example, W=10, D=5) present no challenge, while extremely dense configurations (for example, W=95, D=65) cease to be passable mazes at all. Densities of about W=88 and D=9 seem to give the most satisfactory results. At these densities, some of the mazes produced will be ridiculously easy to get through, while some others will be impassible. Most will be somewhere in between.

The blank line just above the bottom of the frame avoids the frustration of finding a wall or dike sitting right on top of the treasure, and sealing it in. With the blank line, the treasure is always accessible from some part of the maze.

Try experimenting with different densities for the walls and dikes. Try using different symbols for the parts of the maze.



James H. Parsons, 2575 Eastcleft Dr., Columbus, OH 43221.

Typical Run of "Walls and Dikes"

The player starts the program running by pressing RUN and NEWLINE and the display on the screen says:

WALLS AND DIKES
WALL DENSITY?

The player enters a number from 1 to 100, indicating the percentage of space to be filled in by solid horizontal walls.

DIKE DENSITY?

The player enters a number from 1 to 100 indicating the desired percentage of space to be filled in by dikes.

The screen goes blank for a few seconds, and then a maze appears. The maze is framed on all four sides, but there is a gap in the top part of the frame for the player to "enter" by, and a dollar sign in the bottom part of the frame—the "treasure." The location of the door and treasure is determined randomly, as is the distribution of walls and dikes, once their densities have been set. Under the frame is printed:

NEWLINE

When the player presses NEWLINE, the maze is replaced by:

WALL DENSITY

The game begins again. □

```
  10 PRINT ,"WALLS AND DIKES"
  20 PRINT "WALL DENSITY?"
  30 INPUT W
  40 PRINT "DIKE DENSITY?"
  50 INPUT D
  60 CLS
  70 LET A$=CHR$(128)
  80 LET B$=CHR$(0)
  90 GO SUB 1000        (Makes the top of the frame.)
 100 FOR C=1 TO 8       (100-250 generates 8 wall line/dike
                        line pairs.)
 110 PRINT A$;          (Left side of frame for dike lines.)
 120 FOR K=1 TO 21      (120-160 generates dike lines.)
 130 LET E=RND(100)
 140 IF E<D THEN PRINT A$;
 150 IF NOT E<D THEN PRINT B$;
 160 NEXT K
 170 PRINT A$           (Right side of frame for dike lines.)
 180 PRINT A$;          (Left side of frame for wall lines.)
 190 For G=1 TO 21      (190-230 generates wall lines.)
 200 LET H=RND(100)
 210 IF NOT H<W THEN PRINT B$;
 220 IF H<W THEN PRINT A$;
 230 NEXT G
 240 PRINT A$           (Right side of frame for wall lines.)
 250 NEXT C
 260 PRINT A$;          (Left side of frame for blank line.)
 270 FOR L=1 TO 21      (270-290 generates the blank line.)
 280 PRINT B$;
 290 NEXT L
 300 PRINT A$           (Right side of frame for blank line.)
 310 LET B$="S"         (For treasure in bottom of frame.)
 320 GO SUB 1000        (Makes the bottom of the frame.)
 330 PRINT "NEWLINE"    (Reminds player how to get a replay.)
 340 INPUT Z$           (Player hits NEWLINE.)
 350 CLS
 360 GO TO 20
1000 PRINT A$;          (Keeps door/treasure out of left corners.)
1010 LET A=RND(21)      (Chooses location of door/treasure.)
1020 FOR B=1 TO 21
1030 IF NOT B=A THEN PRINT A$;
1040 IF B=A THEN PRINT B$;   (Prints the door/treasure.)
1050 NEXT B
1060 PRINT A$           (Keeps door/treasure out of right corners.)
1070 RETURN
```

Endangered Species

Thirty-six different issues and not one has yet learned to breed in captivity. Their days in our warehouse are limited. Could you provide a good home for a magazine? In exchange, you're assured hours of reading pleasure. And, before they disappear, find the missing link in your own collection.

The applications, programming techniques, simulations, problems, commentary, articles and fiction are practically timeless. Not only that, but the earlier issues are actually increasing in value.

Prices are $2.50 each for all issues after March 1980, $2.00 each for all other issues. $5.00 for three, $15 for ten. Postage is $1.00 for up to 3 issues, $2.00 for 4 or more. Best bet is the **Super Special**: One of everything we have—36 magazines in all—for only $50 post paid.

creative computing

39 E. Hanover Avenue Morris Plains, NJ 07950

Vol. 3, No. 4—July/Aug 1977

Four Dynamic Games: Eliza (in Basic!), Bocce, Backgammon and UFO. First Part of Dwyer's "8-Hour Course in Basic." Writing Your Own CAI: "Structuring the Lesson to the student." Interview with Lee Felsenstein, Designer of the SOL (and several machines since!). Computers in Medicine and Health Care. Evaluations: Spacewar and TREK 80. "Sherlock Holmes and Charles Babbage."

Vol. 4, No. 4—Jul/Aug 1978

Features on Business Computing and Word Processing; Special Section on Interfacing Your Computer to the Outside World: Three Perspectives on Video Games: ROM Section: High Resolution Graphics for Apple II: GAMMON and EVILK Motor Cycle Jump Game Programs. Evaluations: Pet: Apple II: Atari Video Pinball: Atari Video Computer.

Vol. 3, No. 5—Sept/Oct 1977

Dynamic Debugging System for 8080 Assembly Language: Bibliography of "Limits to Growth" Models: Dwyer: 8 Hour Course in Basic—Part 2: Programming Approaches to Solving Complex Equations: Computerized Conferencing. Symmetric Art on your Computer. Games: Nomad, Rotate, Lissajous. Evaluations: 5 Microcomputer Basics: Software Technology Music System.

Vol. 4, No. 5—Sept/Oct 1978

Educational Features: 4 Simulation Articles: Accounts Receivable Systems: Real World Games:A Real-Time Clock You Can Build: All about PASCAL: Intelligent Videodiscs: 40 Programming Ideas: ROM Section: Hex and Star War Games. Evaluations: Exidy Sorcerer: Radio Shack TRS-80: Bally Arcade: Speak & Spell and Spelling B: Computalker Speech Synthesizer: Peninsula Pet Cassettes. Merlin Video Interface.

Vol. 3, No. 6—Nov/Dec 1977

Programming Techniques: File Structures: CAI: Multiple Problem Types: Computer History Quiz: Final Exams by Computer: Dwyer: 8 Hour Course in Basic—Part 3. Mastermind II, Othello, and Inorganic Chemistry Programs. Evaluations: Nine Microcomputer-based Toys: Comp IV: S-100 Compatible Kits: TDL Xitan: and Three 8080 8K Basics.

Vol. 4, No. 6—Nov/Dec 1978

Consumer Computers Buying Guide: Critical Path Analysis: Experiment in Teaching Strategic Thinking: ROM Section, Subject Index and File Index in Basic. Programs for Mail Lists. Patterns. Plotting, Corral, Joust, Puzzle, and a Christmas Letter. Evaluations: CP/M Disk Operating Systems: North Star Horizen: Backgammon Computers: Smart Electronic Games and Video Games.

Vol. 4, No. 1—Jan/Feb 1978

Fast Sorting Algorithm: How to Write a Computer Simulation: More File Structures: LOGO: Murphy's Laws: Dwyer: 8 Hour Course in Basic—Part 4. Programs for World Population Model, Biorhythms in Basic and APL, Yahtzee, Van Gam, Kirkof, and Network. Evaluations: 3 Electronic Games: Radio Shack TRS-80: Heath H8 System.

Vol. 5, No. 1/January 1979

Computers and Robots in Fiction: Guidance Counselor System: Survey of Educator's Attitudes: How to Hide Your Basic Program. A Program to Calculate Depreciation for Taxes, and the Space Maze Game. Counterfeit Cursor and Speed Reading for the Pet. Evaluations: Microsoft Fortran 80: Structured Programming with Tiny c: Smoke Signal's Text Editor: Exidy Sorcerer: Ohio Scientfic Superboard II.

Vol. 5, No. 2—February 1979

Multiple Regression Analysis Simplified: Budget Management: Sports Predictions: PEEKing and POKEing for Video Displays: Interview with Michael Shrayer: Computers and Education—Questions of Value: Game Programs for Gold Mine and Atom 20. Evaluations: Heathkit H-8: Thinker Toys Floppy Disk: Electric Pencil: Western Digital Pascal Chip Set: Four Computer Music Records.

Vol. 5, No. 3—March 1979

Six Articles on Data Base Management: Sports Judging on a Microcomputer: Shopping for a Payroll System: Programming the Game of Go: Business Computing with the Sorcerer: Social Science Survey Program. Evaluations: Terrapin Turtle: Videobrain: Pet Monitor: TRS-80 Floppy Disk: Apple Floppy Disk.

Vol. 5, No. 4—April 1979

Safeguarding Your Computer: Interpretive Programming: Elements of a Good Computer Game: Music Composition: Marin Computer Center. Programs for An Intelligent Calendar, Vertical Graphs and Bar Graphs, Flowers for the PET. Evaluations: Checker Challenger: Video Checkers: Checkbook Maintenance System: Whatsit Data Base Management Program.

Vol. 5, No. 5—May 1979

Word Processing Systems: Pilot Tutorial: Writing User-Oriented Programs: Amoritization Schedules, Reading and Comprehension Exams: Hiding Your Basic Program: Cribbage and Mille Bornes Game Programs. Evaluations: WP Daisy Word Processing: Wordmaster Text Editor: PDI IQ Builder: Malibu 160 Line Printer

Vol. 5, No. 6—June 1979

Eignt Articles on Computer Graphics and Plotting: Using Basic Strings: Microcomputers in the Hospital: Billing Program for the Sorcerer. Inkblot and Greed Game Programs. Evaluations: TRS-80 Voice Synthesizer: HIPLOT Digital Plotter: Structured Systems, Name and Address Program: ALF/Apple Music Synthesizer.

Vol. 5, No. 7—July 1979

Four features on Sorting, Files and Data Bases: Creativity Test: World Power Systems: Personal Finance Model: Two Ecological Simulations: Programs for an Ecological Game, Niche, Brain Teaser, and Zone X. Evaluations: BrighterWriter, SWTPC CT-82 Graphics Terminal: APF PeCos One: Heuristics Speechlab: Micro Pro Super Sort: Diagnostic Programs for the Pet.

Vol. 5, No. 8—August 1979

Can Computers Think?: 5 Basic Language Programming Techniques: The Law and Your Computer: muMath: Image Processing: Manipulating Pencil Files: Adventure, a new type of computer game simulation. The Games HVOLT and FORT. Evaluations: Texas Instruments 99/4: Radio Shack TRS-80 Model II: SWTPC PR-40 for the Pet: IMSAI VIO.

Vol. 5, No. 11—November 1979

Adventure: Complete Listing in Basic. Controlling Household Devices: Car Pooling: Mumps language: Computer Art Exhibition: Build Your Own Joysticks: Telephone Dialer for TRS-80 or Northstar: Teacher-Made Tests: Evaluations: Comparison Chart of 6 Popular Personal Computers: Comparison of 26 Single Board Computers: Electronic Games & Toys: Quick Printer II: Interact Computer: User-Definable Character Generators: TRS-80 Level III Basic: Pet Software from Creative Software: Word Processor: Introl X-10 Home Control System.

Vol. 5, No. 12—December 1979

Controlling Household Devices: Part 2: LOGO: Computerized Biofeedback: Computers at the Rodeo: Creating Digitized Video Images. Programs for using the Microcomputer as an Investment Tool: Animation on the Apple. Magic Tricks. "Turn-Key" CP/M System. Evaluations: More Electronics Games: Language Translators: APF MP1000 Video Game System: 6 Word Processing Printers: Satellite Tracking Software: SysKit for the 8080: Assemblers: CP/M vs. TSC: Statistics for the TRS-80.

Vol. 6, No. 1—January 1980

Interviews with Donald E. Knuth and William Wulf: Six Features on Artificial Intelligence: Air Traffic Controller: Computerized Resume: GROW: A Program that Learns: Evaluations: Six Basics: NEWDOS and TRSDOS: Auto Scribe: Micro Music.

Vol. 6. No. 2—February 1980

Six Articles on Investment Analysis: David Levy: Intelligent Computer Games: Programs: Geneology. Graphing. Genetics: Evaluations: Word Star vs Electric Pencil: Pascal for the TRS-80: Micro Composer: Data Dubber: Sorcerer Word Processing Pac: Trivia Contest Results.

Vol. 6, No. 3—March 1980

Networks for Personal Computers: Artificial Intelligence: How to Make a Basic Tree: Interview with Joel Birnbaum: Three Mile Island Game: Extended Precision Computation: 'Rolodex' Data Base Program: Photographing Your Computer System. Evaluations: TI 99/4. Modems. Cobol. Tiny c. 10 Software Packages.

Vol. 6, No. 4—April 1980

Dr. KiloBYTE's Creative Popular Personal Recreational Micro Computer Data Interface World Journal—the Famous 73 page April Fool parady. 8 Articles on Reading and Language: Interview with Gordon Bell: Evaluations: Heath WH-89: Atari 800 vs PET: Chatworth Mark Sense Card Reader: Adventure.

Vol. 6, No. 5—May 1980

Seven Features on Saving Money with your Computer: Analysis of Stock Options. Budgeting Model. Shopping Lists. Home Inventory. Home Purchase. Retirement Planning. Computer-Aided Model Rocket Design: Two Natural Language Systems: Evaluations: PET 2022 Line Printer: APF Imagination Machine: Personal Software's Desktop Plan: Universal Data Entry System.

Vol. 6, No. 6—June 1980

Fourteen Graphics Articles: Polar Plots. 3-D Graphics. Animation. Graphic Mazes. Motion Simulation. Inside Space Invaders. 7 Music Articles: Digital Audio. Computer-Aided Sight Reading. Design of a Synthesizer. Digital Enhancement of Old Recordings. Comparison of Printers: Evaluations: The Atari Machine: Neelco's Music Box for the PET: HeathKit-Thomas Electronic Organ Kit.

Vol. 6, No. 7—July 1980

Four Articles on Adventure Games: Dragon. Dungeon. How to Fit a Large Program into a Small Machine. How to Write an Adventure. 6 Simulation Features: Genetics. Electric Management. Medical. Ecological. Sports. Self-Reproducing Programs: Man-Machine Dialogs: Selecting a Computer Dealer. Evaluations: Super-Text vs. Easy Writer: Mountain Hardware ROM PLUS+: Toolkit for the Pet: Chart Comparing Basics of 8 Popular Computers.

Vol. 6, No. 8—August 1980

Games Features: Computer Bismarck. Knight's Tour. Guess My Animal. Turnablock Game. Fifteen and Hot. Mind Exerciser. Marketing Your Own Program: Computer Graphic Designs: Robotics Conference: Insertion Sort: Stocks and Listed Options. Evaluations: Magic Wand: VisiCalc: Beta-80: Asteroids in Space.

Vol. 6, No. 9—September 1980

Twenty Educational Applications and Features: Language Arts CAI Development. Grading Program. Computers in the Classroom. Asimov: Point of View: How to Heapsort: New Consumer Electronic Products: TRS-80 Shopping List for Schools. Evaluations: Milliken Math Sequences: Exatron Stringy Floppy: EDS Videotape Series "Little Computers—See How They Run": 8 Apple II Software Packages: Educational Packages.

Vol. 6, No. 10—October 1980

Symposium on Actor Languages and Smalltalk: Linked Merge Sort. How to Solve It: 9 New Applications and Games: Election Prediction. The Presidential Campaign. Computer Division Evaluations: OSI C2-4P Computer. TRS-80 Voxbox. Two Text Editors. Five Music Systems. 15 Software Packages. BASEX.

Vol. 6, No. 11—November 1980

Actor Languages and Smalltalk—Part 2: Effective Documentation: Bombproof Data Entry: Interactive Systems and Virtuality: Fast Sorting: Controlled Input in Basic: Loosening Packed Basic. Future of Small Business Computing: Planning Your Diet. Evaluations: Electronic Games: Computer Ambush: TR Copy.

Vol. 6, No. 12—December 1980

A Comparison of Basic Systems: Buying Guides to Personal Computers and Electronic Toys and Games: Legal Protection of Computer Programs: Nuclear Power Plant Simulation: Superman Game: Logic Problem Solver: Interview with the inventor of the ZX80. Clive Sinclair: Alvin Toffler: The Electronic Cottage. Evaluations: Sinclair ZX80. Atari. Apple Pascal. CBM2022 Smart Printer. Pearl. Apple II and Apple II Plus. Pet. TRS-80.

ROM

July 1977

SOL. The Inside Story: Braille and the Computer Video Newspaper: A Chip is Born: The Care and Feeding of Your Home Computer. Digital Foam—the peripheral of the future.

August 1977

The Kit and I. Part I. by someone who's never soldered before: Introduction to the Fundamentals of Computer Memory: Tips for the Do-it-Yourself Hardware Beginner: Binary clocks: APLomania.

September 1977

Xeroxes and other hard copy off your CRT: Payroll Program: How Computers Work: The Kit and I. Part II: or Power to the Computer: CCD's How They Work and How They're Made: A look at PLATO. an Educational Computer System: IBM 5100.

October 1977

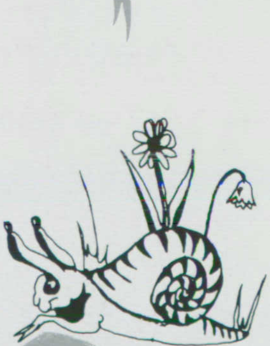
Binary Arithmetic For the Beginner: Microprocessor Aid for the Deaf and Blind: The Kilobyte Card: Scott Joplin on Your Sci-Fi Hi-Fi: Building a Basic Music Board: Flowcharting: Payroll Program.

November 1977

Solar Energy Measurement: A Beginners Introduction to BASIC: The Kit and I. Part III: More Music to Play on Your Computer: Micro Maintenance: Solomon and Viet: Putting Together a Personal Computing System: Time Sharing on the Family MICRO.

December 1977

A Beginners Guide to Peripherals: The Best Slot Machine Game Ever: Artificial Intelligence?: An Electronic Jungle Gym for Kids: File Copy Program: Better Health Through Electronics: The Kit and I Part IV.



January 1978

Synthetic Skin for Your Robot and How to Make It: TLC: A Visual Programming Language: The Code That Can't Be Cracked: Beginner's Guide to Computer Graphics: The Computer and Natural Language: First-Timer's Guide to Circuit Board Etching.

Variable Conversions in the ZX80

Joseph Sutton

There are many cases when it is convenient to convert one type of variable to another, such as numeric to string, or string to numeric. In the ZX80 variable conversions from numeric to string are done with the STR$( ) function. The reverse conversion is not available with the 4K Basic. Given a string variable containing numbers, the program below looks at each number individually to determine its magnitude. The CODES for the numbers go from 28 to 37 (CODE ("0") = 28, CODE ("1") = 29, etc) so that by subtracting 28 from the CODE you get the number itself. By using CODE (A$) - 28
then remove the first character with TL$ ( ).

108 LET A$ = TL$ (A$)

X is then tested with an IF statement to verify that it is a number from 0 to 9, and if it is, it is put into the numeric N with LET N = (N * 10) * X.

109 IF NOT X < 0 AND X < 10 THEN
LET N = (N * 10) + X

Let's assume we have a string containing "123." Line 107 sets X equal to 1 (29-28), then line 108 removes the "1" from A$, and line 109 sets X equal to 1 (N = (0 * 10) + 1). If we return to 106 and start again, A$ = "23", X = 2 and N = 12 (N = (1 * 10) + 2). If we go through a third time, A$ = "3", X = 3 and N = 123 (N = (12 * 10) + 3). Now we need something to test for the end of the string.

113 IF X = -27 THEN RETURN
115 GO TO 107

In the ZX80 all strings end with a "null" character with a CODE of 1. Line 113 tests for this and returns to the main program when the end is detected. Line 115 keeps the routine going until line 113 finds the end. Now, with one more line, we have a Basic subroutine for doing the missing conversion.

```
103 LET N = 0
107 LET X = CODE (A$) - 28
-108 LET A$ = TL$ (A$)
109 IF NOT X < 0 AND X < 10 THEN
    LET N = (N * 10) + X
113 IF X = -27 THEN RETURN
115 GO TO 107
```

This subroutine will search the string A$ until it finds a number. It will then put the numbers into N, stopping when it finds the end of the string, ignoring all other characters. The routine will also destroy the contents of A$, so A$ must be saved if you want it for later use.

With only minor modifications it can be made to accept decimal numbers and scientific notation. First N becomes an ARRAY containing the integer portion in N (0), the fractional portion in N (1), and the exponent (for scientific notation) in N (2). To use this routine as a subroutine all three portions of the ARRAY must be set to zero at the beginning. S is the counter to tell the ZX80 in which part of the ARRAY to place the numbers. We also have to add lines to detect decimal points and "E" if scientific notation of the form 1.5 E 3 is required ($1.5 * 10^3$). Line 110 is added to detect minus signs and lines 116 and 117 properly locate the minus signs. When the fractional portion of the number is processed, the leading zeros are removed. If they are not kept track of, errors will occur; 1.005 would become 1.5, etc. Line 114 detects leading zeros and Z equals the number of leading zeros. In line 30 of the main program you will notice the expression TL$ (STR$ (10 ** Z)). IF Z = 2 (2 leading zeros), 10 ** Z = 100 and the expression becomes the string "100" with the "1" removed or "00". This replaces the zeros in the print statement. No [illegible] ur leading zeros can be [illegible] 5 = 100,000 and y[illegible] error message 6, arithmetic overflow). □

```
10 INPUT A$
20 GOSUB 100
30 PRINT N(0);".";TL$(STR$(10**Z));N(1);"*10**";N(2)
40 STOP

100 LET Z=0
101 LET Y=0
102 DIM N(2)
103 LET N(0)=0
104 LET N(1)=0
105 LET N(2)=0
106 LET S=0
107 LET X=CODE(A$)-28
108 LET A$=TL$(A$)
109 IF NOT X<0 AND X<10 THEN LET N(S)=(N(S)*10)+X
110 IF X=192 THEN LET Y=Y+S+1
111 IF X=-1 THEN LET S=1
112 IF X=14 THEN LET S=2
113 IF X=-27 GO TO 116
114 IF X=0 AND S=1 AND N(1)=0 THEN LET Z=Z+1
115 GO TO 107
116 IF Y=1 OR Y=4 THEN LET N(0)=N(0)*-1
117 IF Y=3 OR Y=4 THEN LET N(2)=N(2)*-1
118 RETURN
```

This is the full subroutine with all the fancy things added and a small "main" program.

ENTERING	PRINTS ON SCREEN
1.414E-3	1.414*10**-3
150	150.0*10**0
-500.8E6	-500.8*10**6
-5.003E-9	-5.003*10**-9

Joseph Sutton, 170 S. Hillside Ave., Succasunna, NJ 07876.

Graphics, Games and Gold

Martin Oakes

Games can be divided into three broad classes. The first has a clearly defined play area which remains the same each time the game is played. Chess, backgammon, tic-tac-toe, and all the card games fall into this category. The second class requires a random area which regenerates differently each time the game is played. Into this group fall the adventure, hunt and seek classics. The last class encompasses all the animated games involving flight and shooting, such as Star Wars and Space Invaders, which require real-time inter-action.

I decided to write my own game using as much of the ZX80 graphics as possible, but without waiting for the 8K Basic ROM and 16K RAM expansion to become available. My choice of game was influenced by several considerations. Everyone knows what the successful outcome of a chess., backgammon, or card game should be, so there is little room for invention. The game either meets expectations or fails. Since the computer is to be a player, it must be a worthy opponent. A dynamic game would have to wait until the 8K ROM becomes available because the screen goes blank during computation with the 4K Basic.

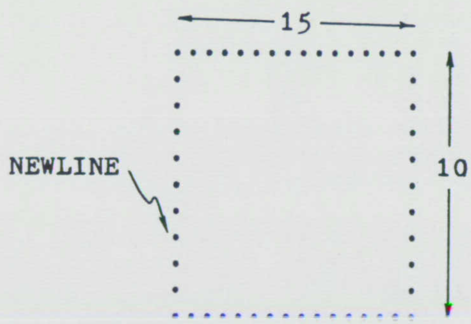


Figure 1. The Game Area.

```
  10  CONSTANTS
 100 MAIN PROGRAM
 200   -CONT-
 300 SQUARE GAME AREA
 400 RANDOM WOOD
 500 PEEK IN DISPLAY
 600 POKE IN DISPLAY
 700 RANDOM PATH
 800
 900 MOVE CHARACTER
1000 SEARCH FOR CHARACTER
1100
1200 FIND AND REPLACE
1300 GOOD OR EVIL?
```

Figure 2. Organization of Program.

My choice then was to plan an "adventure" type of game, which has the added bonus that the writer can make his own rules.

From the beginning I did not expect to fit all the features I wanted into 1K of memory. So instead I wrote a series of subroutines which could be independently de-bugged and set aside to wait for the arrival of more memory. At that time they would be at joined together to make a working program.

The hero is to roam within a randomly generated wood, cave, or castle. In the final version the monsters and treasures he encounters may appear as drawings. The game area is a rectangle 15 characters or columns by 10 lines. Later it can be expanded to occupy as much of the screen as required. See Figure 1.

Each feature of the game is written as a subroutine starting at a line number which is a multiple of 100. See Figure 2.

Martin Oakes, 2100 Oriole Dr., Freeport, IL 61032.

Our hero is going to begin his journey in a wood:

```
400 FOR L = 1 TO 10
405 LET M = L
410 IF L > 5 THEN M = 10 - L
415 LET A = RND (3) + 5 - M
420 FOR C = 1 TO A
425 PRINT CHR$ (128);
430 NEXT C
435 LET B = RND (3) + 2 + M

440 FOR C = 1 TO B
460 PRINT CHR$ (9);
465 NEXT C

470 LET D = 15 - A - B
475 FOR C = 1 TO D
480 PRINT CHR$ (128);
485 NEXT C
490 PRINT
495 NEXT L
```

RUN this program.

We now have a grey wood surrounded by a black border. Each time this is run it is generated differently. We now add randomly dispersed clearings where our hero will find treasures and do battle.

```
445 LET F = 9
450 LET E = RND(12)
455 IF E = 10 THEN LET F = 0
460 PRINT CHR$ (F);
```

Note that line 460 is replaced. RUN this part. Now let's make it into a subroutine called from a main program.

```
100 GO SUB 400
299 STOP

497 RETURN
```

RUN this.

The Jan/Feb 1981 issue of *SYNC* describes on p. 23 how to use the memory address stored in D-FILE to locate the display file.

```
600 POKE ( PEEK (16396) + PEEK
    (16397) * 256 + Q ) . T
605 RETURN

101 LET Q = 12
102 LET T = 58
103 GO SUB 600
```

RUN this. The letter U appeared in the top line of the display. Let's change this to place the U at different points. The game area is a matrix of 15 x 10 characters, but because of the NEWLINE character, each line is really 16 characters long.

```
101 INPUT Q
104 GO TO 101
```

RUN this. The program waits for an input. Try each of the following.

```
1(NL)
3(NL)
17(NL)
```

32(NL) Oops! We destroyed the NEWLINE character. Enter two alphabetic characters to exit with error 2/101.

Delete lines 101, 103, 104.

Now we will develop a subroutine to allow our hero to move around within the wood.

```
900 INPUT A$
901 IF A$ = "0" THEN STOP
905 LET Q = P + 16
910 IF A$ = "N" THEN LET Q = P - 16
915 IF A$ = "E" THEN LET Q = P + 1
920 IF A$ = "W" THEN LET Q = P - 1
940 LET T = 58
945 GO SUB 600
970 LET P = Q
975 RETURN

125 GO SUB 900
101 LET P = 152
150 GO TO 125
```

RUN and enter E.N.W.S to get a string of U's. Type 0 to exit from line 901.

Now we want to erase the trailing (old) positions of U to leave only one in the display.

```
950 LET T = 9
955 LET R = Q
960 LET Q = P
965 GO SUB 600
970 LET P = R
```

RUN this.

Our hero must be confined to the wood until he has earned the right to move on to other adventures.

```
925 IF Q ( 1 OR Q ) 160 THEN GO TO
975
930 GO SUB 500
935 IF NOT T = 9 THEN GO TO 975
```

Look in location Q, to which we will move from the present position P.

```
500 LET T = PEEK (PEEK (16396) +
PEEK (16397) * 256 + Q)
505 RETURN
```

RUN and try to move our hero into a clearing or out of the wood.

Exit with 0(NL).

This subroutine searches for a specific character on a line and replaces it.

```
1200 LET P = L * 16
1205 FOR C = 1 TO 15
1210 LET Q = P + C
1215 GO SUB 500
1220 IF T = S THEN GO TO 1235
1225 NEXT C
1230 RETURN
1235 LET P = Q
1240 LET T = U
1245 GO SUB 600
1255 RETURN
```

Our hero is fated to be cast randomly into the wood to begin his journey.

```
105 LET L = RND (10)
110 LET S = 9
115 LET U = 58
120 GO SUB 1200
```

RUN this.

Now that we have some working subroutines we can set them aside and delete them from memory to make space for new ones.

We will work with a less fancy wood, so delete lines 400-497, and substitute:

```
300 LET F = 9
305 FOR L = 1 TO 10
310 FOR C = 1 TO 15
315 IF RND (10) = 10 THEN LET F = 0

320 PRINT CHR$ (F);
325 LET F = 9
330 NEXT C
335 PRINT
340 NEXT L
345 RETURN
```

For the purpose of checking out the next subroutines, our hero can begin at the bottom of the game area, so delete lines 1200-1255.

Simplify the main program to read:

```
100 GO SUB 300
101 LET P = 152
125 GO SUB 900
150 GO TO 125
299 STOP
```

RUN this and exit with 0(NL).

Our hero cannot enter a clearing in the wood, but he needs to know when he has found one. For this we need a search for a neighboring character routine.

```
1000 LET Q = P - 16
1005 GO SUB 500
1010 IF T = S THEN GO TO 1060
1015 LET Q = P + 16
1020 GO SUB 500
1025 IF T = S THEN GO TO 1060
1030 LET Q = P + 1
1035 GO SUB 500
1040 IF T = S THEN GO TO 1060
1045 LET Q = P - 1
1050 GO SUB 500
1055 IF T = S THEN GO TO 1060
1060 RETURN
```

When our hero finds the clearing, we will replace it with an inverse X.

```
1300 LET T = 189
1305 GO SUB 600
1310 RETURN
```

Since a clearing is a blankspace, S = 0. Add to the main program:

```
102 LET S = 0
130 GO SUB 1000
135 IF T = S THEN GO SUB 1300
```

RUN this program and move our hero around with N, S, E, or W.

We can do something more interesting when our hero finds a clearing. This routine POKEs a random number into the clearing representing gold, which our hero collects.

```
1302 LET V = V + X
1303 LET T = X + 28
1305 GO SUB 600
1310 RETURN

104 LET V = 0

901 IF A$ = "0" THEN GO TO 980
980 PRINT "GOLD", V
985 STOP
```

As you RUN this and move our hero around, he collects the gold. When 0(NL) is typed, his treasure is displayed. At this point we have all the rudiments of an adventure game. From here we can use our imagination to change the options in subroutine 1300. □

Are you in SYNC?

If not, you should be. We would like any programs, translations of existing programs, games or tips which you have to pass on to fellow Sinclair ZX-80 or Micro-Ace owners. Articles are much more lively if accompained by photos (black and white), diagrams, and illustrations. If you do not have an output printer, please type program listings and carefully check them against the listing on the screen. Sample runs should be included with programs rather than just a description of what the program does. Articles should be typed, double space. Your name and address, with phone number should be on first page; all other pages should be numbered. All submissions should include return postage. Payment ranges from $15 to $40 per printed page.

Please send all submissions to:

SYNC
39 E. Hanover Avenue
Morris Plains, New Jersey 07950

A Trick and a Graphic System

by Keith Comer

The ZX80 version of Basic is missing a few statements you may used to using. Getting those graphic characters to do anything really impressive seems downright impossible. The potential is there, but how do you squeeze it out? Using a simulated DATA statement, I have developed a graphic system to allow easy production of graphic displays.

First, let us consider the pseudo-DATA statement. Suppose you need the values 6,4,9 and 7 for some obscure task. Watch:

```
10 LET A$="6497"
20 LET X=CODE(A$)-28
30 GOSUB 1000
40 LET A$=TL$(A$)
50 IF A$+"" THEN STOP
60 GOTO 20
```

Now, line 10 is the "DATA" statement. The terms are just slammed together in a string (A$).

Line 20 fetches the ASCII code for the first character in the string (which is 34) and by subtracting 28 gets the real number you want (which is 6).

Line 30 goes to the part of the program that is to perform that obscure task. The variable X is holding the value you want.

Line 40 clips off the first data element in order to get at the next one.

Line 50 checks to see if the string is empty, in which case we are done and can STOP.

Line 60 starts the whole thing over again (in traditional reiterative fashion).

Obviously, if you need data with more than one digit per entry, you need to trim the digits off one at a time and reconstruct the number from those digits. If you needed 27, 32, 2, and 23, you would use:

```
10 LET A$="27320223"
20 LET X=(CODE(A$)-28) * 10
30 LET A$=TL$ (A$)
40 LET X = X + CODE(A$) -28
50 GOSUB 10000
60 LET A$ = TL$ (A$)
70 IF A$ = "" then STOP
80 GOTO 20
```

Notice that there are no commas between the data elements and that the 2 has to go in as 02.

Keith Comer, 16889 Nichols St. #d, Huntington Beach, CA 92647.

So how does all this number crunching help you realize those incredible graphic displays you have been dreaming about? Watch again:

```
10 LET A$= "££££££££5E55E5E£BB
BBEEBBBBEBBBBEABBBBEEBBBBE£
PBBBEAA3AA4ABBBBAAA£BBBBAA
A3APBBBEAPBBBAAA"

20 LET X= CODE (A$)
30 IF X=12 THEN PRINT
40 IF X=12 THEN GOTO 80
45 IF X < 38 THEN GOTO 110
50 IF X < 48 THEN LET X=X-36
60 IF X > 47 THEN LET X=X+80
70 PRINT CHR$(X);
80 LET A$=TL$(A$)
90 IF A$ = "" THEN STOP
100 GOTO 20
110 FOR N=1 TO X-28
120 PRINT " ";
130 NEXT N
140 GOTO 80
```

Line 10 is the data, as before.

Line 20 fetches the first (or next) character, as before.

Lines 30 and 40 look for the ASCII code 12, which is the £ symbol. It is used to effect a line feed. (Imagine a cross between an "L" and an "F".) This allows you to proceed down the page without typing spaces to the end of each line.

Line 45 checks for an ASCII code which would indicate a number. The routine at 110 will print that many spaces, up to 9 of course; remember, only one digit per data element. This is mainly a convenience, but I have noticed that many pictures I have done are mostly spaces.

Line 50 and 60 convert the ASCII codes of letters from the data line to ASCII codes of graphic symbols. You can save a few bytes (and forsake clarity) by using:

```
50 IF X < 48 THEN LET X= X-116
60 LET X=X+80
```

Line 70 prints the selected graphic symbol.

Lines 80, 90, and 100 should be obvious.

Lines 110 to 140 are the spacing routine referred to by line 45, a simple FOR-NEXT loop.

Now the system works like this: Turn to page 78 in the ZX80 instruction manual and put the letters A through J next to graphic symbols 2 through 11 (ASCII). Put letters M through V next to symbols 130 through 139. The reason that K and L are skipped is so that K can be converted to ASCII code 128 which is the inverted space or a solid black square.

That completes the program. The system allows the use of all the graphic symbols with equal ease, not just the ones on the keyboard. They are all available with one keystroke of typing each. The result is that the screen can be used with the effect of double the resolution, because symbols are used that access any quarter, or any combination of quarters, of each graphic block.

All you do to construct your picture is to draw what you want on graph paper. Your field is 64 squares wide and 46 squares high.

So get your drawing the way you want it. It helps to fill in the squares of the graph paper "all or nothing" (Figure 1). Then divide the picture up into "four squares." This is where the resolution gets halved into the 32 by 23 screen (Figure 2). I just draw over the original lines of the graph paper with ink of another color, every other line. Then convert each "four square" to its corresponding letter from the graphic symbol chart you have just written (Figure 3). Remember to use the £ sign when you get to the end of a line. Now just string the letters together into a data statement for line 10 like this:

```
10 LET A$="GNE£ACA£DN"
```

The advantage of this system is that the program itself is fairly short so you can store a rather elaborate (and/or big) picture in line 10. It cannot hold enough characters to do a full screen (at least not with 1K) but it will hold more than 200. Since it is storing them as one byte each, I doubt that there is a way to store it any tighter. When you save a program, you have also stored the picture.

A few examples are probably in order:

```
INVADERS: "FBKKBE£KRQRQK£NK
           RQKN£DPDCOC"
          "1BOKPB£MKNKNKA£D
           QKNKRC£BN1N1NB"
HEART:    "FKEOP£KKKKKA£QKK
           KKC£1QKKC£2QC"
FACE:     "2BRNPE£10C3QE£MC5Q£
           R2C1C1DA£A1E1C1E1A£
           Q1S2FCMC£DP1NNCFR£
           1DPE1BR£3NNC"
```

More extreme is the

```
CASTLE:   "£78F£78M£77FKP£65M1F
           KKKP£650E1M1M£550KK
           EM1M6A£65AA1M1MBE3
           FP£65AA101MR40KE£65
           AAOC1M4OKKKE£6BBB
           2ANC2M5A1A£5FA1M2A
           3KM4FA1P£502M2A3KM
           4M2M£4FA2DNNC4M4R2
           DA£4MBBE8MBBBBA3P£
           7A1K20NNP7K1M£7A30C
           2DP6K1M£7A3A4M8M£7
           A3A4M8M£7A3A4M8M£7
           PBBBA4MBBBBBBBBO"
```

If you want to do wider spacing, you can use this routine to get two digit's worth of spaces. Eliminate lines 30 and 40 if you want to save space; just space all the way to the end of each line. Replace lines 110 to 140 with:

```
110 LET Z =(X-28) * 10
120 LET A$ = TL$ (A$)
130 LET Z = Z +(CODE (A$) -28)
140 FOR X = 1 TO Z
150 PRINT " ";
160 NEXT X
170 GOTO 80
```

Using this modification you can "draw" a U.S. MAP "3232323226E12PRNNNNNNNNQC06OK11MC09DNRS03RQC11012ARA01OSO12A12PAPRCM13A17M13Q17R13ME16A14PE14OC15PE13A17QBBBE03FBRNNPA21QBEFR04DK23PR06QA22DA" □

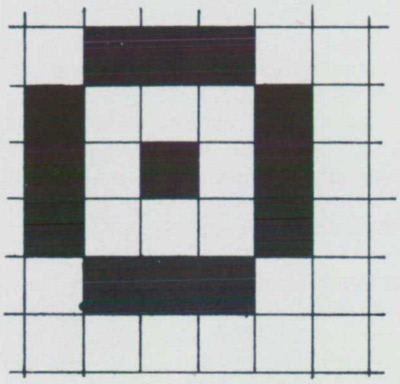


Figure 1.

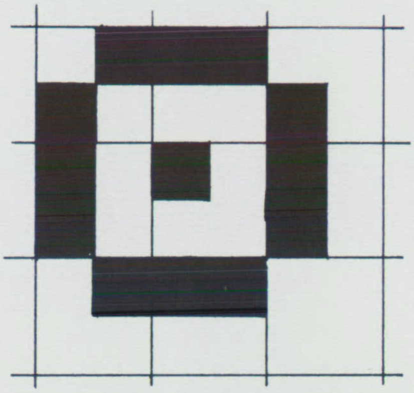


Figure 2.

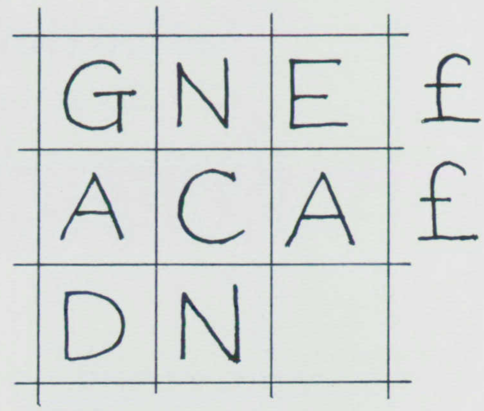
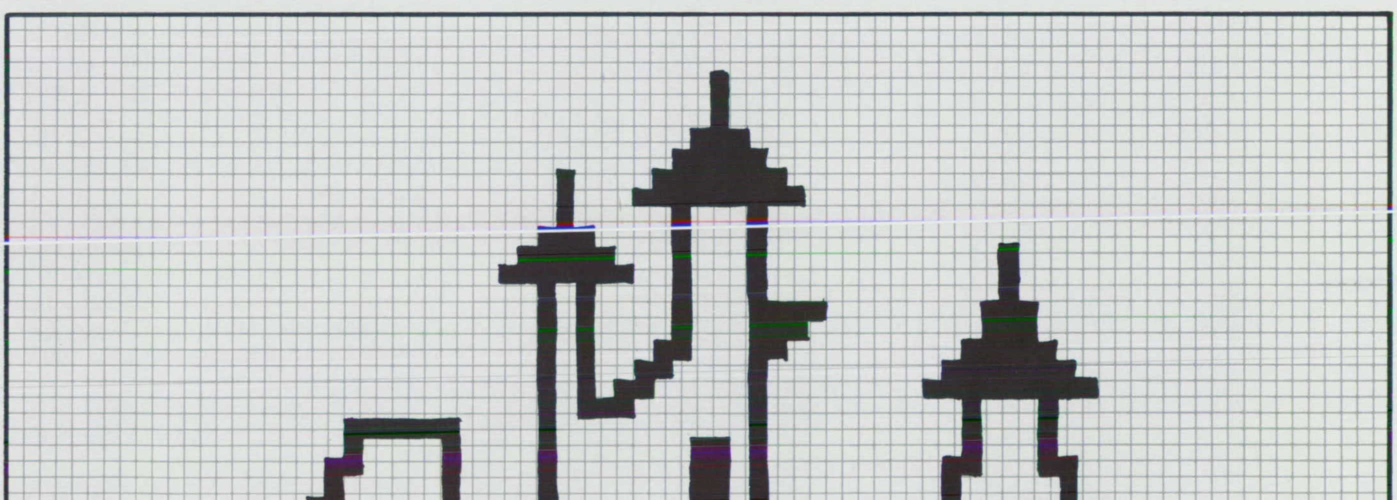
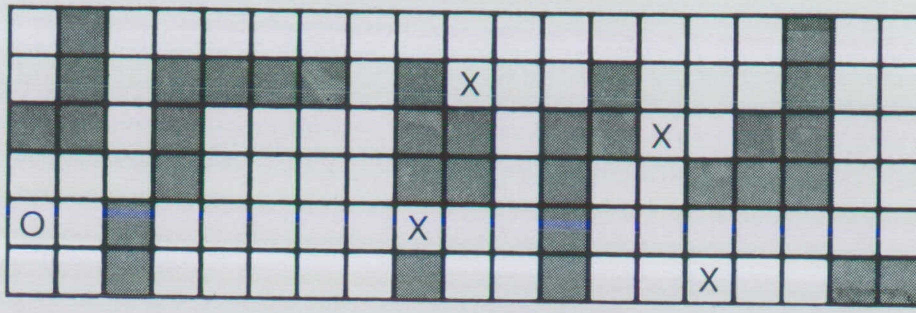


Figure 3.



Sample Output "Castle"



Gauntlet

Ken Berggren

Gauntlet is a game played on a rectangle 19 spaces by 6 spaces. The object is to run a gauntlet of random obstacles and monsters, beginning on the left side and crossing the rectangle. You win when you have successfully moved your marker to the right side.

First you must decide how many monsters you think you can handle, from one to four. Then . . . the screen is randomly sprinkled with blocks. You (0) start at the extreme left. The monsters (X) are between you and your goal, the extreme right.

You move by entering the letters U,D,F or B. For example, to move down and back diagonally you would enter DB or BD. A single letter moves you one space and a Newline alone maintains your position. It is possible to jump an obstacle but if you try to land on one you will not move at all. The monsters frequently blast through the barriers and sometimes that can help you.

The monsters drool green drool, never bathe and have very bad breath. They are also lazy. Except when angry they move only one space at a time. But for all their faults these guys are not dumb. They are very cautious and try to get in front of you before they advance.

When the game is over a NEWLINE will run it again. Any key before the NEWLINE will stop it.

Here are the major sections of the program:

Line 1 is a machine language routine.
Lines 70-120 set the number of monsters.
Lines 130-210 set the starting positions.
Lines 280-430 move the man.
Lines 440-530 move the monsters.
Lines 600-850 end the game or start another.
Lines 900-910 a routine to call the routine.

The machine language routine finds the first character in the display file. It saves

```
  1 REM SPACE
 70 PRINT "HOW MANY MONSTERS?(1-4)"
 80 INPUT N
 90 LET C=0
100 LET D=0
110 CLS
115 IF N>4THEN GO TO 70
120 DIM B(N)
130 FOR L=1 TO 6
140 FOR K=1 TO 20
170 PRINT CHR$(-(RND(8)>5)*9);
180 NEXT K
190 PRINT
210 NEXT L
220 FOR J=1 TO N
230 LET B(J)=RND(L-1)*K-2-(K/5)*(J-1)
240 NEXT J
250 LET G=K*3
260 GO SUB 900
270 GO TO 430
280 LET S=G
290 INPUT C$
300 FOR J=1 TO 2
310 LET C=CODE (C$)
320 IF C=58 THEN LET S=S-K
330 IF C=41 OR S<0 THEN LET S=S+K
340 IF C=43 OR C=39 AND S=(S/K)*K THEN LET S=S+1
350 IF C=39 THEN LET S=S-1
360 IF S>(L-1)*K THEN LET S=S-K
370 IF S+1=((S+1)/K)*K THEN GO TO 700
380 LET C$=TL$(C$)
390 NEXT J
395 GO SUB900
400 IF PEEK(S+D)=61 THEN GO TO 600
405 POKE D+G,0
410 IF PEEK(S+D)=0 THEN LET G=S
430 POKE D+G,52
440 FOR J=1 TO N
450 POKE D+B(J),0
460 LET C=K
470 IF B(J)/C=G/C THEN LET C=1
480 IF B(J)-C<0 THEN LET C=-C
490 IF PEEK(D+B(J)-C)=9 AND RND (9)>4 THEN LET C=0
500 LET B(J)=B(J)-C
510 IF PEEK(D+B(J))=52 THEN GO TO 600
515 IF RND(9)>7 THEN GO TO 460
520 POKE D+B(J),61
530 NEXT J
540 GO TO 280
600 PRINT "GOTCHA"
610 GO TO 800
700 PRINT "YOU ARE FREE"
800 GO SUB 900
810 POKE D+G,20
840 INPUT C$
850 IF C$="" THEN RUN
900 LET D=USR(16427)
910 RETURN
```

Subroutine Loader

```
100 POKE 16403,100
110 FOR J=1 TO 5
120 INPUT C
130 POKE 16426+J,C
140 PRINT J,C
150 NEXT J
```

DECIMAL	LISTING
1	42
2	12
3	64
4	35
5	201

Ken Berggren, 104 Ridgeway Ave., Louisville, KY 40207.

about ten bytes over PEEKing and, with only 1K, every little byte counts.

To load the routine, enter REM and five spaces. Then type in the "loader program" and run it. Enter the five numbers from the "decimal listing" and double check them when you are through. Then type in the main program. You will find that various letters appear around the place newlines are entered. This can be ignored. The condition will disappear when the program lines replace the subroutine loader.

Some of the values for the routine are not character codes, and they do strange things when the ZX80 tries to put them on the screen. Some codes will crash a program. To play it safe push the REM statement off the screen with more program lines or use POKE 16403,100. *Then do not use LIST without a line number*, at least not until you have the program on tape. I do not like that, but I have not found another way to protect a routine and still be able to save it with a program. Any suggestions?

Here are some ways to tailor the program to your own tastes. The TO value in line 140 determines the length of the lines that form the gauntlet. The TO value in line 130 determines the number of lines or the height of the gauntlet. You can change the shape of the display by adjusting those values. However, in 1K this program allows only about 125 characters in the display file. The size of the display file will roughly equal the length of a line plus one, times the number of lines. If you get an error number 4 or 5, it is probably because your display file is too large. To adjust your starting position, change the constant in line 250. A zero starts you on the top line. Adding one to the constant drops you down one line. Be sure that this constant is less than the height of the gauntlet! Finally, if you want to be able to move farther in each turn, increase the TO value in line 300. A three lets you move like a knight in chess. More than three and you are practically unbeatable. □

SYNC Reader Survey

In our first issue of *SYNC* we asked you to tell us about yourselves so that we will be able to make *SYNC* the magazine you want. Your response to our survey has been very positive and enthusiastic. This is what you told us.

First, you told us that you did not like to have surveys printed on the other side of pages you want to keep!

Next, we found that for four out of five of you the ZX80 (or MicroAce) is the only computer you own. About half of you are having your first computer experience with the ZX80. Many admitted yielding to the desire to have a personal computer because of the low price. This enabled you to break into the computer field without making a heavy investment in equipment before you were sure that computers would be a part of your personal activities. The other half have access to a computer at work or at school.

Topping the request list for *SYNC* content is a strong desire for programming tips (four out of five). So if you have a program to submit, remember that your fellow *SYNC* readers are clearly having great fun with their ZX80s, but they are also very eager to learn how to get the most out of their machine. They see every program as a learning opportunity. You will have their deepest appreciation if you share what you have learned about programming through notes in which you point out special tips and explain the main elements in your program.

A close second in requests is for new product information. While we make every effort to find out about new products, our advertisers *and readers* are the main sources of information. If you have found a new product that helps you with your ZX80, please tell the seller, distributor, or manufacturer about *SYNC* so that we can get the news around.

About 75% of our readers want to know more about interfacing techniques and to have software tutorials. Hardware tutorials, graphics software, device control, hardware evaluations, and software evaluations are in the "very much" column for about 60%. Educational, mathematics, and business software are lower on the list with about 40%. While games make the "very much" column for 40%, "very much" and "okay" together include 90% of our readers.

Fiction, puzzles, and advertising came in at the bottom of your list for highest choice, but near the top for your second choice.

For most of our readers additional memory tops the list of planned equipment purchases for 90% with disk and printer capabilities next for 60%. *Creative Computing* and *Byte* are the most widely read computer magazines other than *SYNC*.

The age distribution checks show that 14% are under 20; 27%, 21-30; 31%, 31-40; 13%, 41-50; 13%, 51-60; 1% over 60. Males outnumber females 37 to 1.

Of course, all these are averages based on our survey compilations. It is clear from not only the survey, but also your letters that you have an amazing variety of interests involving your ZX80. Even though the survey summary may not show that other readers have the same special interests as you (and your comments mentioned a number of them), we invite you to share what you find with us. Your fellow readers are always looking for new things to do with their ZX80s. You may open up whole new interests, possibilities, and challenges. □

Try This

This column will feature short programs to show off your ZX80, impress your family and friends, and tickle your imagination when *SYNC* arrives at your place. We invite your contributions. Address them to *SYNC*, 39 E. Hanover Ave., Morris Plains, NJ 07950.

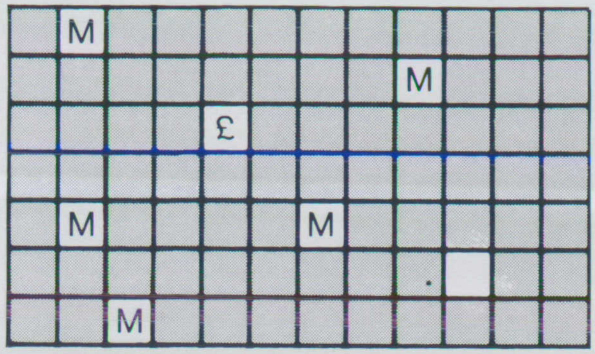
```
10 PRINT CHR$(RND(3));
20 GOTO 10
```

Press RUN and NEWLINE. Disregard the error code which will be displayed. After you have fully absorbed the results of the routine, press any key and then RUN and NEWLINE again. Our thanks to:

Nigel Searle
Sinclair Research Ltd.
50 Staniford St.
Boston, MA 02114 □

Forest Treasure 2K

Paul Frahm




```
10 FOR I=1 TO 20
20 FOR J=1 TO 30
30 PRINT "  ";
40 NEXT J
50 PRINT
60 NEXT I
70 DIM A(50)
80 DIM B(75)
90 LET C=0
100 LET E=RND(20)
110 LET F=RND(30)
120 LET T=RND(619)
130 POKE PEEK(16396)+256*PEEK(16397)+T,0
140 FOR I=1 TO 75
150 LET B(I)=RND(1000)+17300
160 NEXT I
170 LET M=RND(619)
180 POKE PEEK(16396)+256*PEEK(16397)+M,50
190 LET C=C+1
200 LET A(C)=PEEK(16396)+256*PEEK(16397)+M
210 IF A(C)=PEEK(16396)+256*PEEK(16397)+T THEN GOTO 490
220 LET H=F+31*(E-1)
230 POKE PEEK(16396)+256*PEEK(16397)+H,12
240 LET N=PEEK(16396)+256*PEEK(16397)+H
250 FOR I=1 TO C
260 IF A(I)=N THEN GOTO 460
270 NEXT I
280 IF N=PEEK(16396)+256*PEEK(16397)+T THEN GOTO 430
290 INPUT A$
300 IF A$="S" THEN LIST
310 POKE PEEK(16396)+256*PEEK(16397)+H,18
320 FOR I=1 TO 75
330 IF B(I)=N THEN GOTO 400
340 NEXT I
350 IF A$="R" AND F<30 THEN LET F=F+1
360 IF A$="L" AND F>1 THEN LET F=F-1
370 IF A$="D" AND E<30 THEN LET E=E+1
380 IF A$="U" AND E>1 THEN LET E=E-1
390 GOTO 170
400 LET F=F+RND(9)-5
410 LET E=E+RND(9)-5
420 GOTO 170
430 POKE PEEK(16396)+256*PEEK(16397)+T,148
440 PRINT "YOU HAVE TAKEN THE TREASURE"
450 GOTO 500
460 POKE PEEK(16396)+256*PEEK(16397)+H,19
470 PRINT "THE MONSTER HAS KILLED YOU"
480 GOTO 500
490 PRINT "THE MONSTER HAS STOLEN THE TREASURE"%
500 PRINT "ANOTHER GAME? (Y/N)"
510 INPUT B$
520 CLS
530 IF B$="Y" THEN RUN
540 LIST
```

"Forest Treasure" is based upon "Random Graphics" by Gary McGath in the Jan/Feb 1981 issue of *SYNC*. You are riding through a forest, seeking the gold treasure. You are represented by "£" and the treasure is a blank space. During your ride you may encounter enchanted (invisible) walls. When touched, these walls will alter your path, sending you in different directions (sometimes even leaping over the wall!). You may also encounter a monster, represented by the "M" square. This monster has the magical ability to duplicate itself in its search for you. If you run into a monster, or if the monster lands on you, or if the monster steals the treasure, you lose. If you get the treasure, you win. You control your movements by entering U, D, R, or L for up, down, right, or left, and then pressing NEWLINE. Entering S will exit you from the program.

Paul Frahm, 21123 Dettmering, Matteson, IL 60443.

Translating From Other Basics

David Lubar

A command found in many versions of Basic, but not in the Sinclair, is ON ... GOTO. This is usually found in the form ON X GOTO 110, 120, 130. The command makes a jump depending on the value of X. In this example, if X is 1, the program will jump to 110, if X is 2, control goes to line 120, and if X is 3, the program continues at 130. If X is outside the expected values, the program will fall through to the next line. In other words, for any value N of X, the program will jump to the Nth line listed in the expression.

The simplest way to replace this command is to use a series of IF...THEN statements. The above example is equivalent to

```
10 IF X = 1 THEN GOTO 110
20 IF X = 2 THEN GOTO 120
30 IF X = 3 THEN GOTO 130
```

If there are many numbers involved, this process can get tedious. Fortunately, there are other ways to Sync the cat. The Sinclair allows for the use of expressions with a computed GOTO. For example, the above command can be replaced with GOTO 100 + 10 * X. In many cases, you can renumber a translation so the lines used in ON...GOTO will be part of a simple progression. But there are cases where the progression is not simple.

Take a line such as ON X GOTO 90, 450, 376, 10. Rather than look for an algorithm that will produce the correct number, it is easier to set up an expression. Using the logical capabilities of the Sinclair, we can produce an expression that has the desired sum for any X value. What we need is a series where the sum of each member is zero unless it matches the desired X value. When there is a match, the sum will be the value of the desired line for the jump. The above line can be replaced with GOTO ABS ((X=1) * 90 + (X=2) * 450 + (X=3) * 376 + (X=4) * 10). This expression will produce the desired results. Those parts of the expression where the equality fails will produce a value of 0. When there is a match, the result will be correct except for having a negative value. This is caused by the use in the Sinclair of -1 to signify true. The ABS takes care of that.

Another common Basic operation is the LEN function. The expression LET X = LEN (A$) will give X a value equal to the number of characters in A$. If A$ is HELLO, then X will be 5. This expression has many uses. Once you know the length of a string, you can manipulate it in various fashions. While the Sinclair does not have the LEN function, it does have TL$ which removes the first character of a string. Using TL$ in a loop, the length of any string variable can be determined. The basic approach is to keep chopping off the first character of a string until there is nothing left. If you count how many beheadings have occurred, you will know the length of the string. Here's one way to do it.

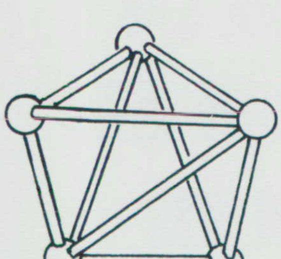
```
10 INPUT A$
20 LET L = 0
30 LET B$ = A$
40 IF B$ = "" THEN GOTO 100
50 LET B$ = TL$ (B$)
60 LET L = L + 1
70 GOTO 40
100 PRINT A$; "HAS A LENGTH OF" ; L
```

The program is fairly straightforward. Since TL$ destroys the variable, A$ is preserved by using B$ for the operation. When B$ has only one character left, the result of TL$ (B$) will produce a null string (represented in line 50 as a pair of quotes with nothing between them).

That's all for now. If you have any specific functions you would like to see covered here, drop me a line.

puzzle answers

A Building Problem:



Lucky Number: Multiply the selected number by 9, and use the product as the multiplier for the larger number. It will be found that the results will be respectively as under:

12345679	x 9 = 111	111	111
"	x 18 = 222	222	222
"	x 27 = 333	333	333
"	x 36 = 444	444	444
"	x 45 = 555	555	555
"	x 54 = 666	666	666
"	x 63 = 777	777	777
"	x 72 = 888	888	888
"	x 81 = 999	999	999

It will be observed that the result is in each case the "lucky" number, nine times repeated.

The Puffer-Belly Problem: The speed of the two trains in relation to one another is 45 + 36 = 81 miles per hour. This equates out to:

$$\frac{5{,}280 \times 81}{60 \times 60} = 118.8 \text{ feet per second}$$

The length, then, of the Dover train is 6 x 118.8 = 712.8 feet.

A Seven-Letter Charade: The answer is the word ENGLAND. The other words are END, GLAD, ANGEL, LAND.

The Three Jealous Husbands: For the sake of clearness, we will designate the three husbands *A*, *B*, and *C*, and their wives *a*, *b*, and *c*, respectively. The passage may then be made to the satisfaction of the husbands in the following order:

1. *a* and *b* cross over, and *b* brings back the boat.
2. *b* and *c* cross over, *c* returning alone.
3. *c* lands and remains with her husband, while *A* and *B* cross over. *A* lands, *B* and *b* return to the starting point.
4. *B* and *C* cross over, leaving *b* and *c* at the starting point.
5. *a* takes back the boat and *b* crosses with her.
6. *a* lands and *b* goes back for *c*.

The Four Jealous Husbands: Distinguishing the four husbands as *A*, *B*, *C*, and *D*, and the four wives as *a*, *b*, *c*, and *d*, respectively, the answer to this version is:

1. *a*, *b*, and *c* cross over; *c* brings back the boat.
2. *c* and *d* cross over; *d* brings back the boat.
3. *A*, *B*, and *C* cross over; *C* and *c* bring back the boat.
4. *C*, *D*, and *c* cross over.
5. *c* takes back the boat and fetches *d*.

8K Basic ROM and 16K-Byte RAM Pack Specifications

The 8K Basic ROM and the 16K-Byte RAM pack are now available from Sinclair Research (see Resources column). The specifications for these units are as follows.

ZX80 8K BASIC ROM

The 8K Basic ROM for the ZX80 is designed for high-level, full-facility computing. The chip—a drop-in replacement for the existing 4K Basic ROM—comes with a new keyboard template and a supplementary operating manual.

Key features of the new 8K BASIC ROM include -

- fully floating-point arithmetic to 9-digit accuracy,
- logs, trig, and their inverse functions,
- graph plotting facility,
- animated displays using PAUSE n,
- full set of string-handling facilities,
- n dimensional arrays,
- n dimensional string arrays,
- cassette LOAD and SAVE with named programs.

Full specification follows.

Numbers

Stored in 5 bytes in floating point binary form giving 9×10^{-44} to 1.1×10^{43} accurate to 9 1/2 decimal digits.

Variables

Numeric:	Any letter, followed by alphanumerics.
String:	A$ - Z$.
FOR-NEXT:	A - Z.
Numeric arrays:	A - Z.
String arrays:	A$ - Z$.

Arrays

Numeric arrays:	'n' dimension, subscript range starts at 0.
String arrays: (more correctly, character arrays)	'n' dimension, subscript range starts at 0. If the last subscript is omitted it's treated as a fixed length string.

Strings

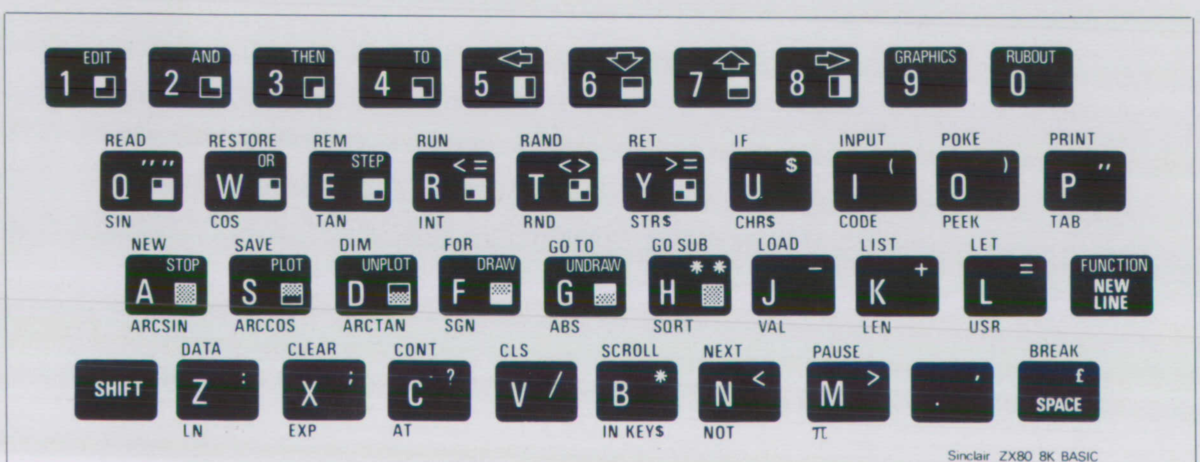
Undimensioned strings can be any length.
Can be concatenated (+).
Substring eg B$ = A$ (2 TO 4).
Literal strings eg C$ = "QWERTY".

Statements available

In this list,

v	represents a variable.
x,y,z	represent numerical expressions.
m,n	represent numerical expressions that are rounded to the nearest integer.
e	represents an expression.
f	represents a string valued expression.
s	represents a statement.

Note that arbitrary expressions are allowed everywhere (except for the line number at the beginning of a statement). Thus "GOTO LN A ** 2" is valid.



Keyboard template for new 8K BASIC ROM.

CLEAR	Deletes all variables, freeing the space they occupied.
CLS	(Clear Screen) deletes all PRINT output in the display file.
CONTINUE	Resumes execution of the last run program—repeats the last statement if an error was detected, otherwise restarts at the next one. Note that a command (immediate execution) statement counts as a program and so destroys the re-entry data.
DATA...	Standard, but no unquoted strings.
DIM...	Deletes any array or string with the same name, sets up space for a new array in the usual way, and initialises its element to 0 or " ".
DRAW m,n	Let (u,v) be the current PLOT (q.v.) position. Draws a line as straight as possible from (u,v) to (u + m, v + n) by blacking in pixels (quarter character squares). Changes the PLOT and PRINT positions.
FOR A TO B STEP C	Generally standard, but entirely dynamic in its action.
NEXT	The effect of a NEXT statement is to look up the corresponding FOR-variable, increment its value by the STEP, check whether the limit is exceeded and if not jump to the looping line number.
GOSUB n	Transfers control to BASIC subroutine.
GOTO n	Jumps to line n.
IF x THEN s	If x is true (defined to mean greater in absolute value then 2^{-112}) then s is executed. The standard values of true and false as yielded by relational operators are 1 and 0.
INPUT v	Outputs the display file to the screen with no special INPUT prompt; the rest is standard. Cannot be used as a command (immediate execution) statement.
LIST	Lists from start of program.
LIST n	Lists program starting at line n with program cursor pointing at line n.
LOAD f	Looks for a program called f on tape and loads it and its variables.
NEW	Default n = 0. Erases BASIC program and variables.
NEW n	n is used to alter a system variable known as RAM TOP, which is the address of a byte in RAM. The area from RAM TOP on is untouched by the BASIC system, and POKEd programs can be left there in safety.
PAUSE n	Sends the display file to the TV screen for n frames (50 frames per second) or until a key is pressed.
PLOT m,n	Sends the PLOT position (a system variable) to (m,n) and blacks in that pixel. Also changes the PRINT position.
POKE m,n	Writes n in byte m in RAM.
PRINT...	Mostly standard. The display file has 22 lines of 32 characters each (2 zones of 16 characters) and when this is filled it is sent to the TV with error 5. CONTINUE carries on with the program with no loss of data.
PRINT AT m,n	Moves the PRINT position to line m, character n.
PRINT TO de	Alters the PRINT format. Here d is an optional digit between 1 and 8 (default value 8) and e is an optional letter E. From now until another such formatting item, numbers will be printed to d significant digits, and if E is present they will always be printed using scientific notation. On switch-on, the format is initialised so that numbers are printed to 8 digits and scientific notation is avoided where possible. Note that PRINT does not change the PLOT position.
RANDOMIZE	Standard
RANDOMIZE n	If n is given this is made the value of the seed of the random number generator.
READ v	Reads v from a data statement.
REM...	Remember, for program comments.
RESTORE	Reinitialises the data (so it can be read again).
RETURN	Return from subroutine.
RUN	RUNs the BASIC program.
RUN n	CLEAR followed by GOTO n.
SAVE f	Saves program and variables on tape and calls it f.
SCROLL	Scrolls display file up one line, losing top line and making space at bottom.
STOP	
UNDRAW m,n UNPLOT m,n	These are like DRAW and PLOT, but blank out pixels instead of blacking them in.

Functions	Type of Operand	Result
	number	Negate
ABS	number	Absolute magnitude
ARCOS	number	In Radians
ARCSIN	number	In Radians
ARCTAN	number	In Radians
CHR$	number	The character whose code is x.
CODE	number	The code of the first character is x (or 0 if x is empty)
COS	number	In radians.
EXP	number	e^x.
INKEY $	number	Reads the keyboard. The result is a character representing the key pressed, otherwise the empty string.
INT	number	Integer.
LEN	string	The length of x.
LN	number	Natural log
NOT	number	Exclusive—ORs the first byte of x with 113, so that NOT 0 = 1, NOT 1 - 0. Unlike the other functions, NOT has binding power 4 (between AND and the relational operators) NOT A = B has the same value as NOT (A=B) (and A< >B).
PEEK	number	The value of the byte in store whose address is x.
PI		π (3.1415927)
RND		A random number between 0 and 1.
SGN	number	Yields -1, 0, +1.
SIN	number	In Radians.
SQRT	number	Square root.
STR$	number	The string of characters that would appear on the screen if x were PRINTed.
TAN	number	In Radians.
USR	number	Converts x to an address in store and calls that address as a machine code subroutine. On return, the result is the contents of the h1 register pair.

Functions	Type of Operand	Result
VAL	string	Evaluates x as a numerical expression (x must not contain the quote image character).
AND		Logical AND
OR		Logical OR

Relational operators

=	Equal
>	Greater than
<	Less than
<=	Less than or equal to
=>	Greater than or equal to
<>	Not equal

Graphics

All characters, their reverses, and all graphics can be entered directly from the keyboard.

ZX80 16K-BYTE RAM PACK

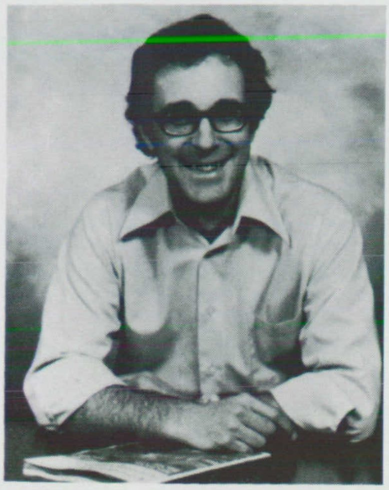
The complete module is designed to provide massive add-on memory capacity.

The 16K-BYTE RAM pack can be used for program storage or as a database. Yet it costs up to half the price of competitive additional memory.

Measuring approximately 3" x 3" x 1.25" the RAM pack plugs into the existing expansion port on the rear of the Sinclair ZX80 via an edge connector. No additional power supply is needed. □

"This one is called 'Kafka'. It is programmed to try to present the user from figuring out how to play it."

A REMARKABLE MAGAZINE



David Ahl, Founder and Publisher of Creative Computing

creative computing

"The beat covered by Creative Computing is one of the most important, explosive and fast-changing."—Alvin Toffler

You might think the term "creative computing" is a contradiction. How can something as precise and logical as electronic computing possibly be creative? We think it can be. Consider the way computers are being used to create special effects in movies—image generation, coloring and computer-driven cameras and props. Or an electronic "sketchpad" for your home computer that adds animation, coloring and shading at your direction. How about a computer simulation of an invasion of killer bees with you trying to find a way of keeping them under control?

Beyond Our Dreams

Computers are not creative per se. But the way in which they are used can be highly creative and imaginative. Five years ago when *Creative Computing* magazine first billed itself as "The number 1 magazine of computer applications and software," we had no idea how far that idea would take us. Today, these applications are becoming so broad, so all-encompassing that the computer field will soon include virtually everything!

In light of this generality, we take "application" to mean whatever can be done with computers, *ought* to be done with computers or *might* be done with computers. That is the meat of *Creative Computing*.

Alvin Toffler, author of *Future Shock* and *The Third Wave* says, "I read *Creative Computing* not only for information about how to make the most of my own equipment but to keep an eye on how the whole field is emerging.

Creative Computing, the company as well as the magazine, is uniquely lighthearted but also seriously interested in all aspects of computing. Ours is the magazine of software, graphics, games and simulations for beginners and relaxing professionals. We try to present the new and important ideas of the field in a way that a 14-year old or a Cobol programmer can understand them. Things like text editing, social simulations, control of household devices, animation and graphics, and communications networks.

Understandable Yet Challenging

As the premier magazine for beginners, it is our solemn responsibility to make what we publish comprehensible to the newcomer. That does not mean easy; our readers like to be challenged. It means providing the reader who has no preparation with every possible means to seize the subject matter and make it his own.

However, we don't want the experts in our audience to be bored. So we try to publish articles of interest to beginners and experts at the same time. Ideally, we would like every piece to have instructional or informative content—and some depth—even when communicated humorously or playfully. Thus, our favorite kind of piece is acessible to the beginner, theoretically non-trivial, interesting on more than one level, and perhaps even humorous.

David Gerrold of *Star Trek* fame says, "*Creative Computing* with its unpretentious, down-to-earth lucidity encourages the computer user to have fun. *Creative Computing* makes it possible for me to learn basic programming skills and use the computer better than any other source.

Hard-hitting Evaluations

At *Creative Computing* we obtain new computer systems, peripherals, and software as soon as they are announced. We put them through their paces in our Software Development Center and also in the environment for which they are intended—home, business, laboratory, or school.

Our evaluations are unbiased and accurate. We compared word processing printers and found two losers among highly promoted makes. Conversely, we found one computer had far more than its advertised capability. Of 16 educational packages, only seven offered solid learning value.

When we say unbiased reviews we mean it. More than once, our honesty has cost us an advertiser—temporarily. But we feel that our first obligation is to our readers and that editorial excellence and integrity are our highest goals.

Karl Zinn at the University of Michigan feels we are meeting these goals when he writes. "*Creative Computing* consistently provides value in articles, product reviews and systems comparisons . . . in a magazine that is fun to read."

Order Today

To order your subscription to *Creative Computing* send payment to the appropriate address below. Customers in the continental U.S. may call toll-free to charge a subscription to Visa, MasterCard or American Express.

Term	*USA*	*Canada and Foreign Surface*	*Foreign Air*
1 year	$20	$29 or £12.50	$50 or £21
2 years	$37	$55 or £24.00	$97 or £41
3 years	$53	$80 or £34.50	$143 or £61

We guarantee your satisfaction or we will refund your entire subscription price.

Join over 80,000 subscribers like Ann Lewin, Director of the Capital Children's Museum who says, "I am very much impressed with *Creative Computing*. It is helping to demystify the computer. Its articles are helpful, humorous and humane. The world needs *Creative Computing*."

creative computing

P.O. Box 789-M
Morristown, NJ 07960
Toll-free **800-631-8112**
(In NJ 201-540-0445)

27 Andrew Close, Stoke Golding
Nuneaton CV13 6EL, England

Key Click Generator

by Matthew J. Johnson

This simple circuit will produce an audible tone whenever the ZX80 screen is blanked, yielding a click to indicate key closure, or a steady tone during processing.

Examination of the "SYNC" line (IC 19—PIN 5) with an oscilloscope reveals three constituent signals: Line Sync (denoted $\overline{LS}$); Frame Sync ($\overline{FS}$); and Keyboard ($\overline{KBD}$). These signals are low assertion, as indicated, and have period and repetition rates as follows:

Signal	Period	Rep. Rate
$\overline{LS}$	6us	58us
$\overline{FS}$	380us	18ms
$\overline{KBD}$	6ms	N/A

The time constant of the RC low pass filter was chosen to allow the second gate to switch only on signals long with respect to $\overline{FS}$, i.e., $\overline{KBD}$. When SYNC returns high, the first gate sinks the discharge current of the capacitor via the Germanium diode, resetting the circuit. Germanium is used here to insure that the minimum negative-going threshold voltage ($V_{\tau} < 0.6V$) of the second gate is reached, turning off the buzzer. The Schmitt-trigger was chosen for its high positive-going threshold, enabling a less critical circuit design than would be possible using standard TTL.

The buzzer (a piezoelectric job from Radio Shack #273-064) drive circuit is taken directly from the blister pack except for the diode added to PIN 8 to clamp positive oscillations to the five volt rail.

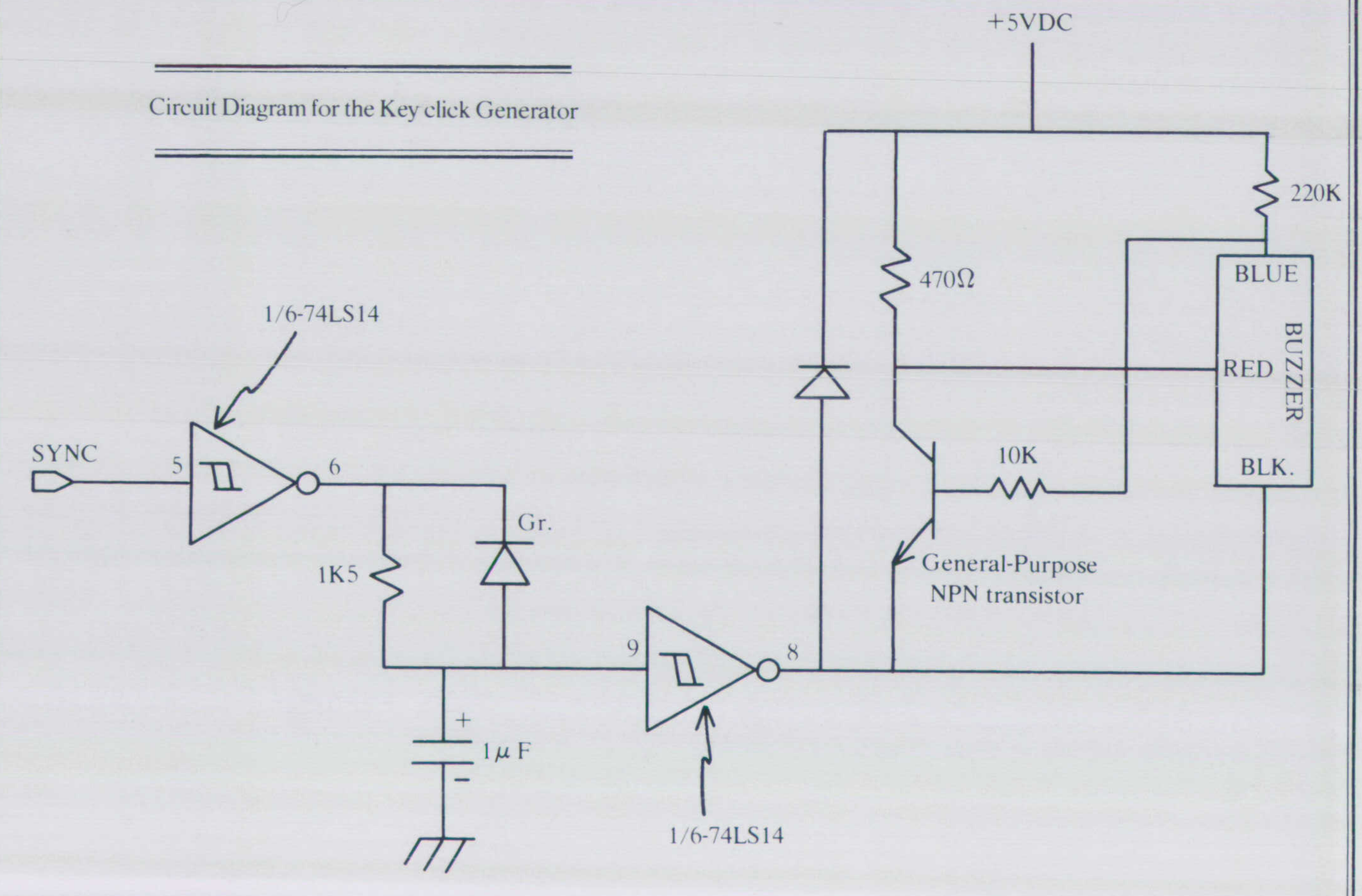
A 74LS121 one-shot could be used instead of the 74LS14 to eliminate the steady tone during processing, but I prefer having the sound as an indication of processing activity.

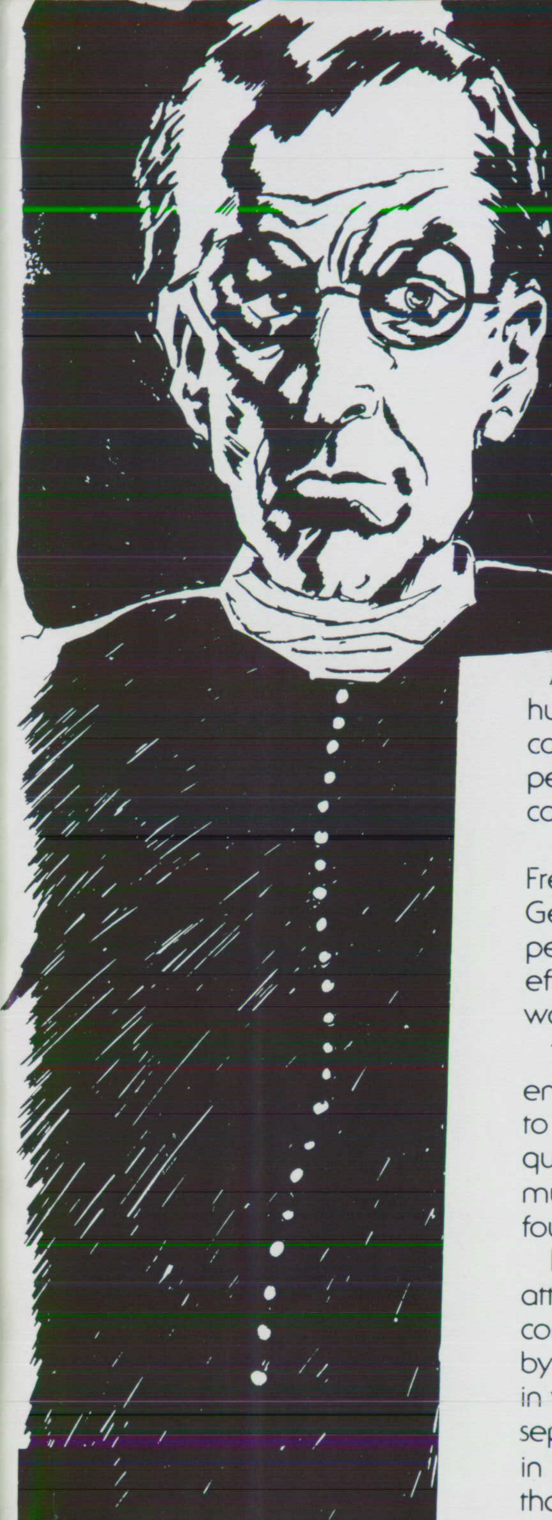
The Schmitt-trigger was "piggy-backed" on IC 19, picking up power and the common (PIN 5) SYNC signal with a dab of solder. I used that "double sided stick 'em stuff" intended for wall hangings to mount the buzzer across two other IC's. The balance of the circuit was "sky-wired" and the entire modification was neatly fitted within the standard ZX80 case, so as not to obtrude in an obvious manner. A bit of insulating tape may be needed on the inside of the cover to prevent shorts.

The audio feedback has made life with the ZX80 passive keyboard enjoyable and has freed my attention to concentrate on what—as opposed to how—I am typing. □

Matthew J. Johnson, 92 Devir St., Malden, MA 02148.

Circuit Diagram for the Key click Generator





Only Fiction . . . or is it?

Tales of the Marvelous Machine:

35 Stories of Computing

A robot friend. A computer God. Artificial intelligence challenging human intelligence in a life and death struggle. A detective solving a computer murder. Computers tricking people or people tricking people with computers. A computer with a soul. Or power. A lonely computer. Or one in love with its operator.

In thirty-five wonderful stories about computers, authors such as Frederick Pohl, Charles Mosmann, M.V. Mathews, Carol Cail, and George Chesbro depict a life in which computers affect the way people live, think, and relate to each other. Interested in what the effect of computer saturation might be? Only fiction can so wonderously dramatize future life.

The book is fun, and will provide wonderful hours of entertainment. For the reader interested in a structured approach to understanding the potential roles of the computer, or wanting quickly to locate stories that support or challenge his viewpoint, a multiple table of contents is provided. This lists the stories in fourteen different categories.

For example, a list of stories in which the computer takes on the attributes of a human separates them from those in which the computer is only an intelligent machine. The stories are categorized by whether they clarify, improve, or worsen the human lot. Stories in which the computers have capabilities available today are separated from those in which the capabilities could be available in the future. There is a listing of the wildly whimsical stories and those in which the computer is utilized in a unique fashion.

Can criminals be caught by computer? Does computer crime pay? Do computers fall in love? Are we all part of a larger organic computer? Here are 35 tantalizing tales that will open your eyes to a new perspective of computers.

Skillfully drawn illustrations augment the stories, giving glimpses of scenes as envisioned by 20 talented artists. This artwork adds another dimension to the text.

Tales of the Marvelous Machines: 35 Stories of Computing, edited by Robert Taylor and Burchenal Green, is a beautiful big 8½" × 11" softbound anthology of 272 pages. 12B

It is available for $7.95 plus $2.00 shipping and handling per order from Creative Computing, P.O. Box 789-M, Morristown, NJ 07960. NJ residents add 5% sales tax. Visa, MasterCard and American Express orders are welcome. For faster service, call in your bank card order toll free to **800-631-8112** (in NJ call 201-540-0445). Or use the handy order form bound into this magazine.

creative computing press

The ZX80 Keyboard *James H. Parsons*

The ZX80's keyboard is of the simple membrane type which is matrix scanned to read a key. The principle behind a membrane keyboard is relatively simple and is illustrated below in Figure 1. The base layer is a printed circuit board which has a matrix of circular contacts, like those shown in Figure 2, laid out in a grid. Each contact has two traces running from it.

The top layer of the system is the flexible keyboard template. Located above each contact on the base layer is a small, circular contact. When a key is pressed, the contact on the bottom side of the template presses down on its respective keyboard contact, creating a conductive path, and thus closing the switch.

The process by which a key closure is located is called *matrix scanning*, and it works as follows. As you will note by looking at the schematic diagram of the keyboard in Figure 3, the rows of the keyboard are connected to the anodes of a group of diodes. The cathodes of the diodes are connected to the higher eight address lines. The columns of the keyboard are connected to the inputs of IC10, a 74LS365 tri-state bus driver. The diodes are used to inhibit sinking of the address lines by the pull-up resistors (R13-R17). The resistors are used just on good design principle and do not make any major functional difference in the machine; in fact, the system works without them.

To scan for a key, sequence through each address line, setting it low and all other high. Read the column data from ICI0. When an address line is low, its respective diode will allow a logic 0 to pass through; when an address line is high, its respective diode will create an output similar to that of a tri-stated output.

James H. Parsons, 1921 Flintlock Terrace West, Colorado Springs, CO 80918.

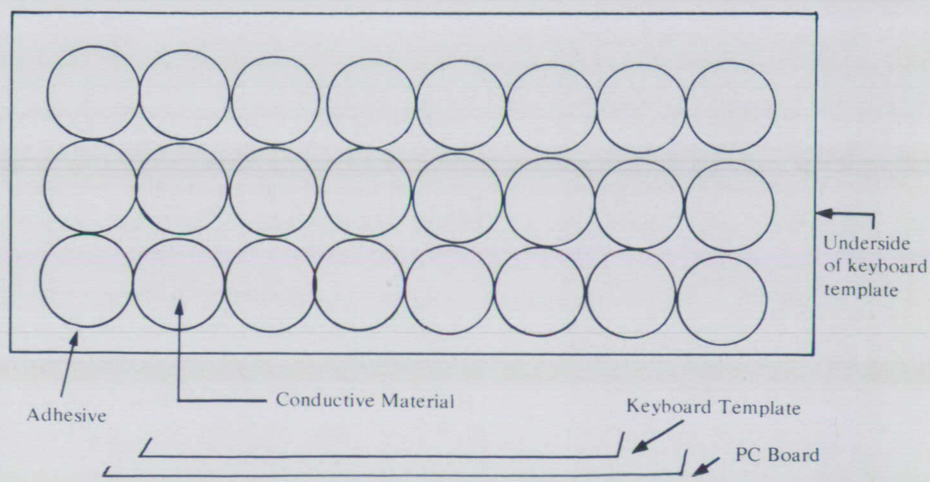


Figure 1.

When a key closure is made, either a low signal or a tri-state signal is sent to the input of IC10. IC10, being a standard 74LSxx gate, has internal pull-up resistors on its inputs. A tri-state type signal presented as input to IC10 will, therefore, allow the pull-up resistor to pull-up the input line and turn the input transistor on, thus causing a logic 1 to be the effective input. When a logic 0 input is received, the input line becomes grounded, and the internal input pull-up is disabled, thus causing a logic 0 to be the effective input.

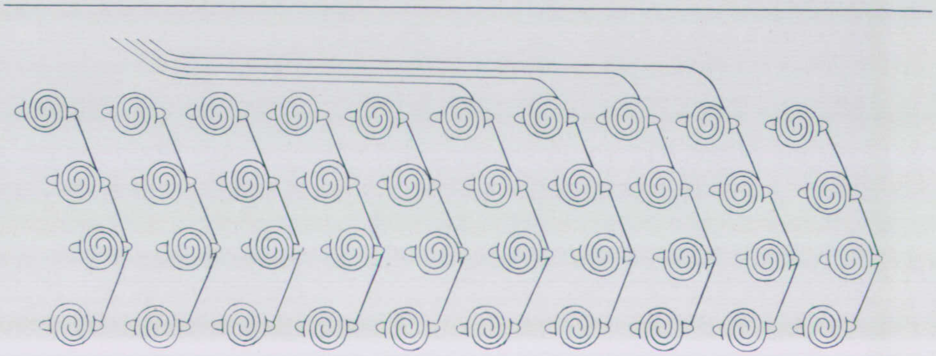


Figure 2.

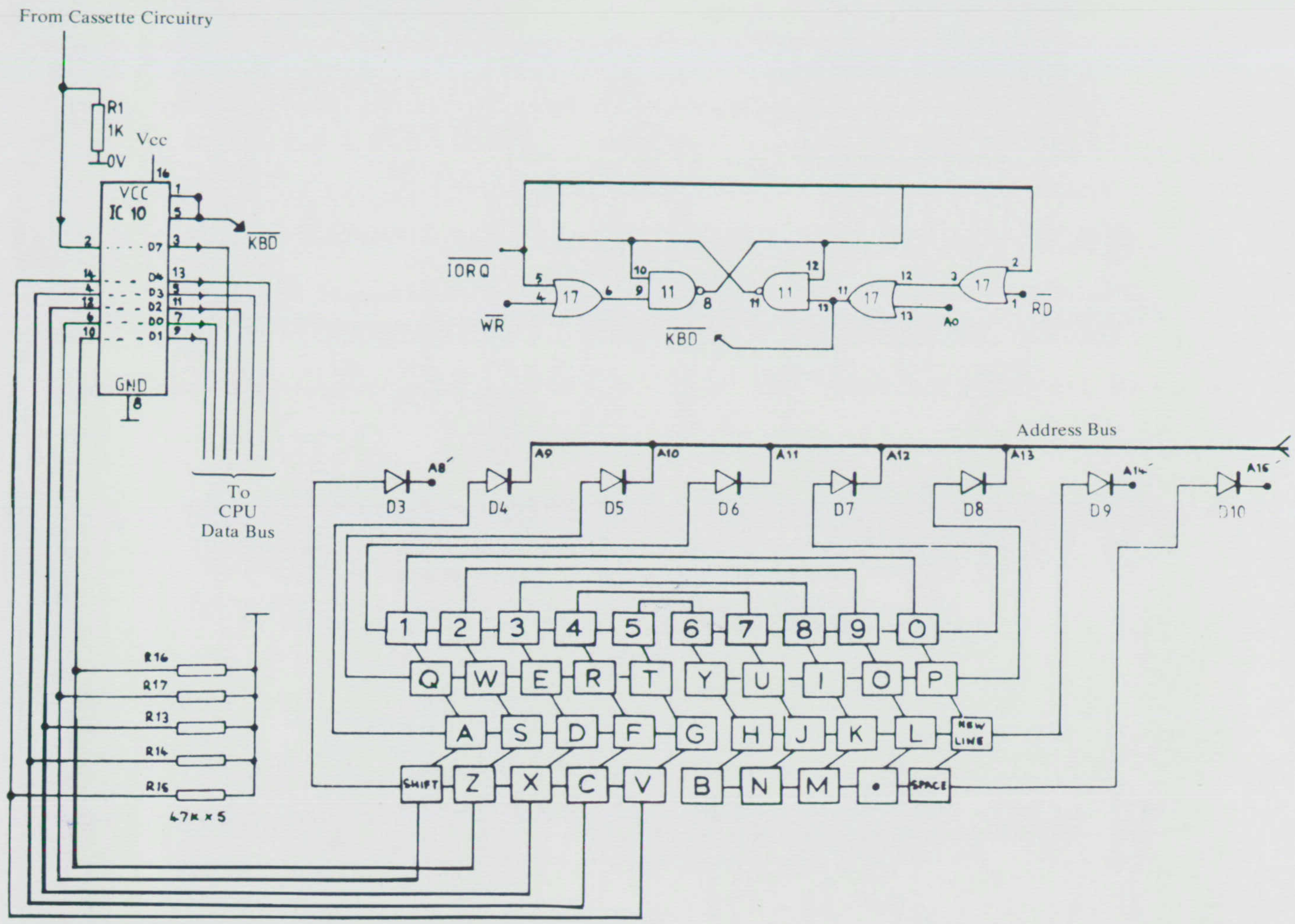


Figure 3.

IC10 is enabled when the signal $\overline{KBD}$ (see Figure 3) is active (i.e., low). As you will note, the signal is derived from two OR gates. Logically, the signal is $\overline{KBD}$ = A0 + $\overline{RD}$ + $\overline{IORQ}$. Essentially, all of the three inputs must be low to enable IC10. This means that an I/O read (a Z80 IN instruction) is being done from any even address (i.e., any address with A0 = 0).

During an I/O request ($\overline{IORQ}$=0), the contents of the A register are placed on the higher eight bits of the address bus. During a keyboard read, the higher eight bits of the address are referred to as the keyboard mask. Executing an IN A, FEh instruction will output the keyboard mask and then read the value of IC10 into the A register. (NB FEh is *not* the only possible port address; any even value will work.)

A simple routine to test for the BREAK key is shown below:

```
LD A,7Fh
IN A, (FEh)
RRA
JR NC, BRKPRS
```

The first instruction loads the keyboard mask into A. This particular mask has all but the ms bit of A (bit 7) set (i.e., 0111 1111 binary). The IN instruction puts out the mask and reads a column from the keyboard. With a mask of 7Fh, the column read is BREAK, EDIT, P, RUBOUT, NOT, NEW, LIST, SHIFT.

When the IN terminates, if no keys were hit, all of the keyboard bits (i.e., d0-d4 of A) will be set. If a key is pressed, then its corresponding bit in A will be a logic 0, provided it was in the selected column. After the IN instruction, the data for the BREAK key will, therefore, reside in bit 0 of A.

The RRA instruction rotates the contents of register A one bit to the right. Bit 7 comes from the data in the carry flag. The carry flag is set to the data in bit 0 position of A (i.e., the data for the BREAK key). Now the carry flag will contain a 0 if BREAK was pressed; otherwise it will hold a 1. The next instruction, if the carry flag is clear, will jump to BRKPRS.

The keyboard and display subroutine scans the keyboard to see if a key was pressed; if not, it passes a frame to the display and loops back to the keyboard scan section. If a key is pressed, then the routine will return to its caller. This routine is shown in Listing 1.

To use the routine, execute a CALL 13Ch instruction. It will return a value in the BC register pair, which corresponds to the keyboard mask and column input for the key pressed. Bits 5, 6, and 7 will be set to ones by the OR 0E0h instruction at 5$:. Bit 0 of B will be zero if SHIFT was pressed; otherwise it will be a 1. C will hold the keyboard mask. For example, if the Z key is pressed, B will hold F7h (i.e., 1111 0111) and C will hold FEh (i.e., 1111 1110).

Listing 2 shows a method for obtaining a ZX80 character in A. The subroutine FILLDF assures that there are enough NewLines in the display file.

I hope that this article has provided some insight into the workings of the ZX80 keyboard. □

Listing 1.

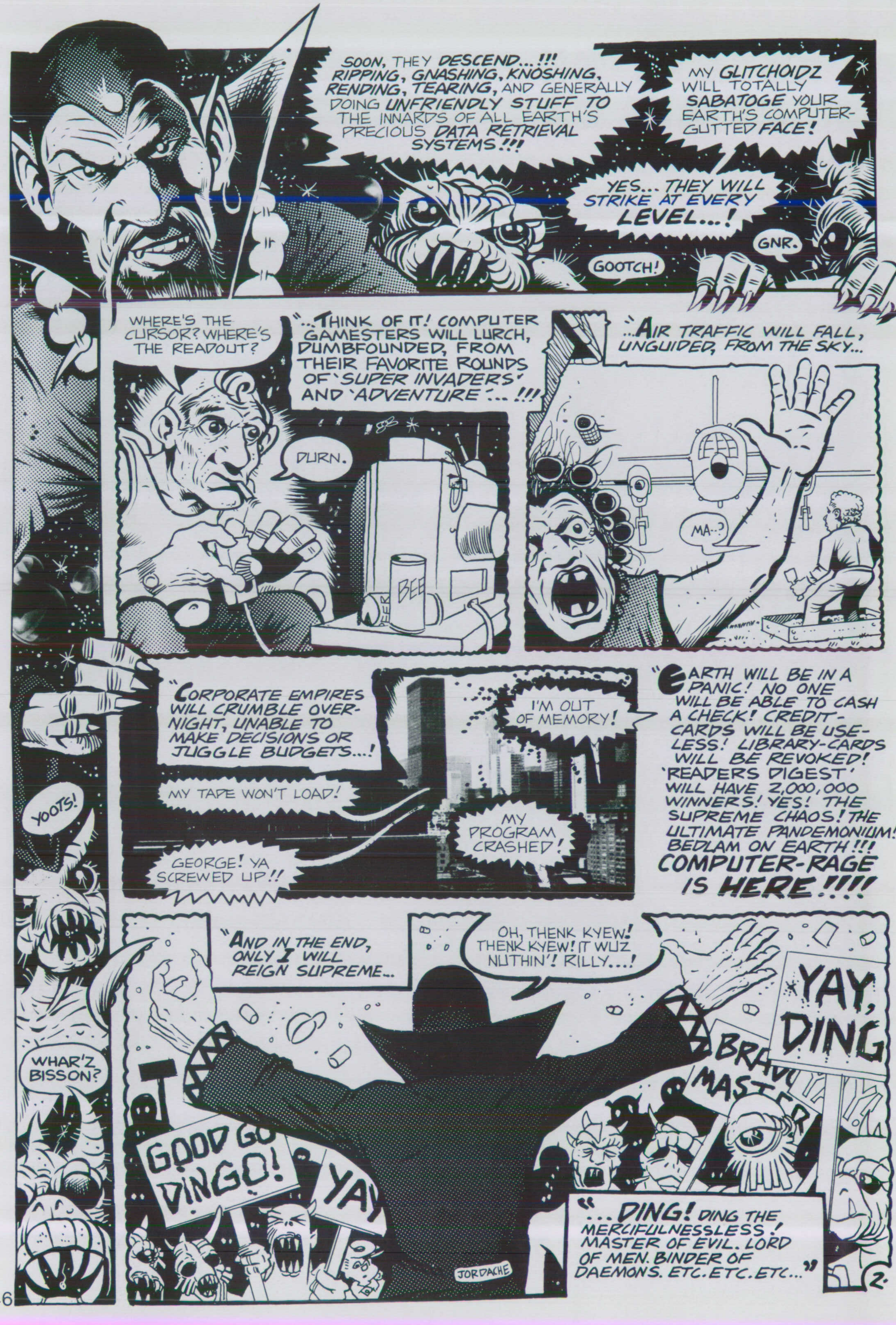
```
RESULT:        EQU     4022h
FRAMES:        EQU     401Eh
CH_ADD:        EQU     4026h
LOOP:   Call Show;     Space between last line of chars
                       and fram sync
DISP:               ;  Enter here from BASIC to get a
                       key and display the current
                    ;  display file
                    ;
                    ;  Address 319 decimal
                    ;              13F hex
    LD B.8
1$  DJNZ $          ;  Blow away 99 T-States
    LD HL. (FRAMES) ;  Get old frame counter
    INC HL          ;  Increment it
    LD (FRAMES).HL  ;  Put it back
    LD HL.-1
    LD B. ØFEh
    LD C.B
    IN A. (C)       ;  Start frame sync
    OR 1
5$  OR ØEØh
    LD D.A          ;  Zero bit for each key pressed
    CPL             ;  Flip bits
    CP 1
    SBC A.A         ;  ØØ if any key pressed. else FFh
    OR B
    AND L
    LD L.A
    LD A.H
    AND D
    LD H.A
    RLC B           ;  Rotate mask left
    IN A. (C)
    JR C.5$         ;  IF Ø in mask hasn't reach carry
    RRA
    RL H
    RLA
    RLA
    RLA
    SBC A.A
    AND 24: Ø if US. 24 if UK
    ADD A.32
    LD (RESULT+1).A ;  32 if US. 56 if UK
                    ;  no L has a Ø for each row in
                       which a key. other than SHIFT.
                    ;  was pressed; H similarly for
                       columns in dl-d5. d6d7 are
                    ;  ones. dØ=Ø. if SHIFT pressed.
                       else dØ=1
                    ;  717 T-States since start of frame
                       sync. 545 before end
    LD BC. (CH_ADD) ;  Pick up last times key hits. or a
                       value with
                    ;  d15d14=Ø1 if first time around
    LD (CH_ADD). HL
    LD A.B
    ADD A.2         ;  Now either carry is clear and BC
                       indicates a key was pressed
                    ;  or carry is set and BC=FFFFh
                       or FEFFh.
                    ;  N.B. Neither 0000h nor FF00h is a
                    ;  possible value for HL. since d6
                       d7 are set and. if all of d1 to d5
                       of H
                    ;  L=-1
    SBC HL.BC       ;  HL:=Ø if HL=BC and C=FFH
    EX DE HL
    LD HL. RESULT
    LD A. (HL)
    OR D
    OR E
    RET Z           ;  If (X_PTR)=BC. a key is
                       depressed and
                    ;  count = Ø. exit with A.D.E=Ø
    LD A.B
    CP 254
    SBC A.A.
    AND B
    RRA
    LD (HL). A
    DEC B
2$: DJNZ 2$
    OUT (ØFFh). A   ;  frame sync ends at next M1
    LD A.-2Ø
    LD B.25
    LD HL. (D_FILE) ;  Get HL= first byte of display
                       file
    SET 7.H         ;  Insure Interrupt
    CALL SHOW       ;  Display space above picture and
                       24
    LD A.-13        ;  lines of text
    INC B
    DEC HL
    DEC (IY + RESULT
    +1-Y)           ;  One less line below picture than
                       above
    JR LOOP
SHOW: LD C. (IY+RESULT
    +1-Y)           ;  #picture lines in first line of text
                       (31)
    LD R.A
    LD A.-35        ;  Value for R in subsequent lines
    EI
    JP (HL)         ;  Will return to caller at end of
                       picture
```

Listing 2.

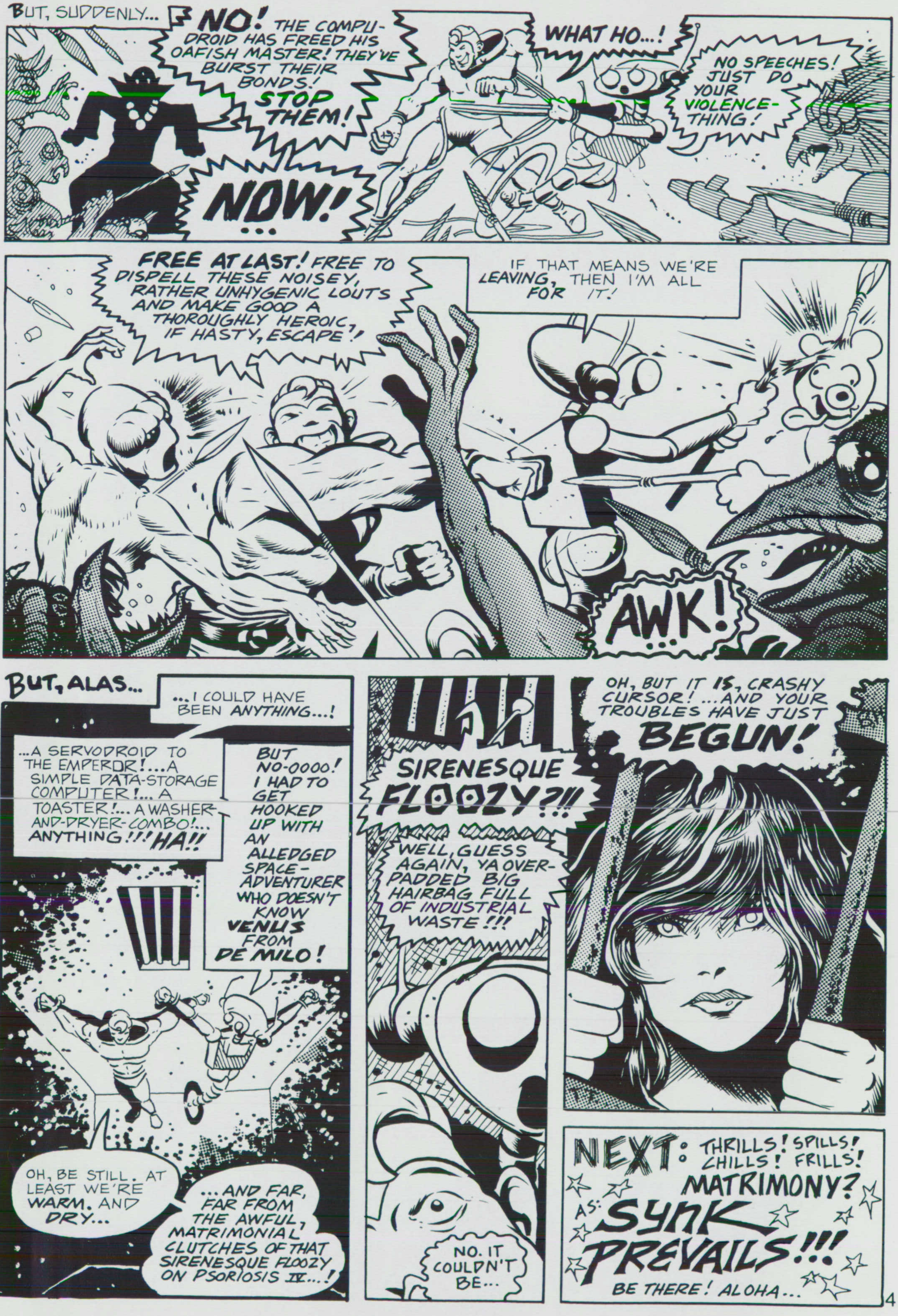
```
KB TAB:  EQU 06Ch
FILLDF:  EQU 05C2h
DISP:    EQU 013Fh
KWLOW:   EQU 0E6h
GETKEY:  Call FILLDP      :  Fill display file w/reqd N/Ls
         Call DISP
         SRA B
         SBC A.A
         OR 38
         LD L.5
         SUB L
    $1:  ADD A.L
         SCF
         RR C
         JR C.$1
         INC C
         JR NZ.GETKEY     :  If more than one bit set
         LD HL.KB_TAB-1
         LD E.A
         ADD HL.DE
         LD A. (HL)
         JR.Z.$2
         ADD A.0C0h       :  Here if in KW state: i.e.. convert
                             from
         CP KWLOW         :  letter to keyword
         JR N C.$2
         LD A. (HL)
    $2:  RET              :  Here with char in A
```

CRASH CURSOR
THAT'S RIGHT, ADVENTURE-FANS! CRASH CURSOR, AND HIS FAITHFUL COMPU-DROID, SYNK, HAVE BEEN BRUTALLY CAPTURED BY THE EVIL DING THE MERCIFULNESSLESS, AND HIS MAD CREATIONS, THE THOROUGHLY NASTY, SOON-TO-BE-HOUSEBROKEN--GLITCHOIDZ...!!
...MORE THAN THAT, WE AINT SAYIN'...!
SO! CRASH CURSOR!!
WE MEET AGAIN! ONLY THIS TIME, NOTHING CAN SAVE YOU!!
UH--
--I COULDN'T INTEREST YOU IN A LITTLE CASH, COULD I, DING, OL' BOY...?
WATCH THAT SPEAR, SHORTY! I'M NOT PROGRAMED TO BE A PIN-CUSHION!!! WHAT'S YOUR GAME, DING?!
MY GAME?!! WHY, AS EVER, ONE OF CONQUEST, DEAR DROID!
YOU ARE ABOUT TO WITNESS MY FINEST HOUR!!!
"...SOON A DIVISION OF MY 'GLITCHOIDZ' (TM) WILL HURTLE INTO YOUR WORLD'S ATMOSPHERE, CARRYING CHAOS AND DISORDER FROM THE STARS...!
"...ONCE THERE, THEY WILL CARRY OUT MY MOST RUTHLESS SCHEME TO DATE...!
..."MICRO-MINIATURIZED, MY LEGIONS WILL INFILTRATE THE COMPLEX CIRCUITRY OF EVERY COMPUTER-SYSTEM ON EARTH, WREAKING HAVOK WHERE-EVER THEY CAN...!"

1



SOON, THEY DESCEND...!!! RIPPING, GNASHING, KNOSHING, RENDING, TEARING, AND GENERALLY DOING UNFRIENDLY STUFF TO THE INNARDS OF ALL EARTH'S PRECIOUS DATA RETRIEVAL SYSTEMS!!!
MY GLITCHOIDZ WILL TOTALLY SABATOGE YOUR EARTH'S COMPUTER-GUTTED FACE!
YES... THEY WILL STRIKE AT EVERY LEVEL...!
GNR.
GOOTCH!
WHERE'S THE CURSOR? WHERE'S THE READOUT?
"...THINK OF IT! COMPUTER GAMESTERS WILL LURCH, DUMBFOUNDED, FROM THEIR FAVORITE ROUNDS OF 'SUPER INVADERS' AND 'ADVENTURE'...!!!
DURN.
...AIR TRAFFIC WILL FALL, UNGUIDED, FROM THE SKY...
MA..?
"CORPORATE EMPIRES WILL CRUMBLE OVER-NIGHT, UNABLE TO MAKE DECISIONS OR JUGGLE BUDGETS...!
I'M OUT OF MEMORY!
MY TAPE WON'T LOAD!
MY PROGRAM CRASHED!
GEORGE! YA SCREWED UP!!
YOOTS!
"EARTH WILL BE IN A PANIC! NO ONE WILL BE ABLE TO CASH A CHECK! CREDIT-CARDS WILL BE USE-LESS! LIBRARY-CARDS WILL BE REVOKED! 'READERS DIGEST' WILL HAVE 2,000,000 WINNERS! YES! THE SUPREME CHAOS! THE ULTIMATE PANDEMONIUM! BEDLAM ON EARTH!!! COMPUTER-RAGE IS HERE!!!!
"AND IN THE END, ONLY I WILL REIGN SUPREME...
OH, THENK KYEW! THENK KYEW! IT WUZ NUTHIN'! RILLY...!
WHAR'Z BISSON?
YAY, DING
BRAV MASTER
GOOD GO DINGO!
YAY
"...DING! DING THE MERCIFULNESSLESS! MASTER OF EVIL. LORD OF MEN. BINDER OF DAEMONS. ETC. ETC. ETC..."
JORDACHE
2.



BUT, SUDDENLY...
NO! THE COMPU-DROID HAS FREED HIS OAFISH MASTER! THEY'VE BURST THEIR BONDS! STOP THEM!
NOW!...
WHAT HO...!
NO SPEECHES! JUST DO YOUR VIOLENCE-THING!
FREE AT LAST! FREE TO DISPELL THESE NOISEY, RATHER UNHYGENIC LOUTS AND MAKE GOOD A THOROUGHLY HEROIC, IF HASTY, ESCAPE!
IF THAT MEANS WE'RE LEAVING, THEN I'M ALL FOR IT!
AWK!...
BUT, ALAS...
...I COULD HAVE BEEN ANYTHING...!
...A SERVODROID TO THE EMPEROR!...A SIMPLE DATA-STORAGE COMPUTER!...A TOASTER!...A WASHER-AND-DRYER-COMBO!... ANYTHING!!! HA!!
BUT NO-OOOO! I HAD TO GET HOOKED UP WITH AN ALLEDGED SPACE-ADVENTURER WHO DOESN'T KNOW VENUS FROM DE MILO!
OH, BE STILL. AT LEAST WE'RE WARM. AND DRY...
...AND FAR, FAR FROM THE AWFUL, MATRIMONIAL CLUTCHES OF THAT SIRENESQUE FLOOZY ON PSORIOSIS IV...!
NO. IT COULDN'T BE...
SIRENESQUE FLOOZY?!!
WELL, GUESS AGAIN, YA OVER-PADDED BIG HAIRBAG FULL OF INDUSTRIAL WASTE!!!
OH, BUT IT IS, CRASHY CURSOR! ...AND YOUR TROUBLES HAVE JUST BEGUN!
NEXT: THRILLS! SPILLS! CHILLS! FRILLS! MATRIMONY? AS: SYNK PREVAILS!!!
BE THERE! ALOHA...

Resources for the ZX80 and MicroAce

We welcome entries from manufacturers and readers for the resources column. Please include the name of the item, a brief description, price, and complete data on how to obtain it. Send contributions to SYNC Resources, 39 East Hanover Avenue, Morris Plains, New Jersey 07950.

Software

- Moving graphics games
  Super ZX80 Invasion (1K and 2K) and *Double Breakout*. Cassettes, $14.95 each plus $1.50 shipping. Check or money order to:
  SOFTSYNC, INC.
  P.O. Box 480
  Murray Hill Station
  New York, NY 10156

- Games and educational software.
  Hardware and technical information in the near future.
  TENSOR TECHNOLOGY
  4 Morning Dove
  Irvine, CA 92714

- ZXBUG (1½K)
  A machine code degugging program; useful for programming in machine code. An annotated disassembled listing of the 4K Basic. Much more software.
  Artic Computing
  396 James Reckett Avenue
  Hull HU8 OHA
  England

- 1K games:
  Adventure Cassette (*Dragonslayer*; *Lunar Landing*; graphics)
  Vegas Cassette (*Black Jack*, *Slots*)
  $6.95 per cassette (postage included); check or money order only.
  J. Schwitalla
  1235 Pickwick Place
  Flint, MI 48507

- Games, subroutines, and teaching aids in Basic and Machine Code. SASE for complete list.
  Zeta Software
  P.O. Box 3522
  Greenville, SC 29608

- 7 Games for the ZX80 and MicroAce (one cassette)
  $11 from:
  New England Software
  Box 691
  Hyannis, MA 02601

- ZX80 Software on cassette. Games, educational, programming course.
  Bug-Byte
  251 Henley Road
  Conventry CV2 1BX

Users Groups

- Educational ZX80/1 Users's Group
  Highgate School
  Birmingham B12 9DS
  U.K.
  (Publishes a newsletter)

- ZX80 Amateur Radio Users' Group
  (for licensed amateur radio operators)
  c/o K2MI, Martin H. Irons
  46 Magic Circle Drive
  Goshen, NY 10924

- National ZX80 Users Club
  Membership free; publishes *Interface* magazine; send large, stamped, addressed envelope plus one 10p stamp to:
  National ZX80 Users Club
  44-46 Earls Court Road
  London, W8 6EJ
  England

Hardware

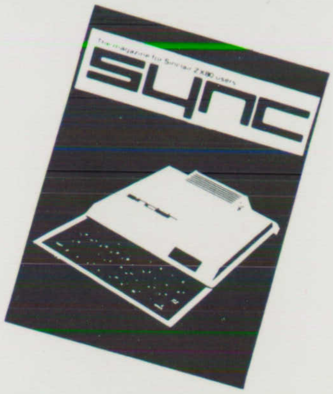
- Super Isolator, Model ISO-11
  A control for severe AC power line spikes, surges, and hash.
  $94.95. Call (617) 655-1532
  Electronic Specialists, Inc.
  171 S. Main St.
  Natick, MA 01760

- 16K-Byte RAM for massive add-on memory.
  $99.95 + $4 shipping.
  8K Basic ROM (replacement chip for ZX80).
  $39.95 + $4 shipping.
  Phone orders (Visa, Master Card):
  800-543-3000, op. 508
  Mail orders:
  Sinclair Research Ltd.
  1 Sinclair Plaza
  Nashua, NH 03061

- Keyboard beeper, $12
  Burnett Electronics
  908 Morris St.
  Cincinnati, OH 45206

- 55 Key keyboards for the ZX80 or MicroAce.
  Complete plans for $5: keyboard $14.95.
  Complete kit (keyboard, parts, etc.): $29.95.
  Schultz Systems
  1026 Ferdinand
  San Antonio, TX 78245

The Sinclair ZX80 is innovative and powerful. Now there's a magazine to help you get the most out of it.



Get in sync

SYNC magazine is different from other personal computing magazines. Not just different because it is about a unique computer, the Sinclair ZX80 (and kit version, the MicroAce). But different because of the creative and innovative philosophy of the editors.

A Fascinating Computer

The ZX80 doesn't have memory mapped video. Thus the screen goes blank when a key is pressed. To some reviewers this is a disadvantage. To our editors this is a challenge. One suggested that games could be written to take advantage of the screen blanking. For example, how about a game where characters and graphic symbols move around the screen while it is blanked? The object would be to crack the secret code governing the movements. Voila! A new game like Mastermind or Black Box uniquely for the ZX80.

We made some interesting discoveries soon after setting up the machine. For instance, the CHR$ function is not limited to a value between 0 and 255, but cycles repeatedly through the code. CHR$ (9) and CHR$ (265) will produce identical values. In other words, CHR$ operates in a MOD 256 fashion. We found that the "=" sign can be used several times on a single line, allowing the logical evaluation of variables. In the Sinclair, LET X=Y=Z=W is a valid expression.

Or consider the TL$ function which strips a string of its initial character. At first, we wondered what practical value it had. Then someone suggested it would be perfect for removing the dollar sign from numerical inputs.

Breakthroughs? Hardly. But indicative of the hints and kinds you'll find in every issue of **SYNC**. We intend to take the Sinclair to its limits and then push beyond, finding new tricks and tips, new applications, new ways to do what couldn't be done before. **SYNC** functions on many levels, with tutorials for the beginner and concepts that will keep the pros coming back for more. We'll show you how to duplicate commands available in other Basics. And, perhaps, how to do things that can't be done on other machines.

Many computer applications require that data be sorted. But did you realize there are over ten fundamentally different sorting algorithms? Many people settle for a simple bubble sort perhaps because it's described in so many programming manuals or because they've seen it in another program. However, sort routines such as heapsort or Shell-Metzner are over 100 times as fast as a bubble sort and may actually use less memory. Sure, 1K of memory isn't a lot to work with, but it can be stretched much further by using innovative, clever coding. You'll find this type of help in **SYNC**.

Lots of Games and Applications

Applications and software are the meat of **SYNC**. We recognize that along with useful, pragmatic applications, like financial analysis and graphing, you'll want games that are fun and challenging. In the charter issue of **SYNC** you'll find several games. Acey Ducey is a card game in which the dealer (the computer) deals two cards face up. You then have an option to bet depending upon whether you feel the next card dealt will have a value between the first two.

In Hurkle, another game in the charter issue, you have to find a happy little Hurkle who is hiding on a 10 X 10 grid. In response to your guesses, the Hurkle sends our a clue telling you in which direction to look next.

One of the most ancient forms of arithmetical puzzle is called a "boomerang." The oldest recorded example is that set down by Nicomachus in his *Arithmetica* around 100 A.D. You'll find a computer version of this puzzle in **SYNC**.

Hard-Hitting, Objective Evaluations

By selecting the ZX80 or MicroAce as your personal computer you've shown that you are an astute buyer looking for good performance, an innovative design and economical price. However, selecting software will not be easy. That's where **SYNC** comes in. **SYNC** evaluates software packages and other peripherals and doesn't just publish manufacturer descriptions. We put each package through its paces and give you an in-depth, objective report of its strengths and weaknesses.

SYNC is a Creative Computing publication. **Creative Computing** is the number 1 magazine of software and applications with nearly 100,000 circulation. The two most popular computer games books in the world, *Basic Computer Games* and *More Basic Computer Games* (combined sales over 500,000) are published by Creative Computing. Creative Computing Software manufactures over 150 software packages for six different personal computers.

Creative Computing, founded in 1974 by David Ahl, is a well-established firm committed to the future of personal computing. We expect the Sinclair ZX80 to be a highly successful computer and correspondingly, **SYNC** to be a respected and successful magazine.

Order SYNC Today

To order your subscription to SYNC, in the USA send $10 for one year (6 issues), $18 for two years (12 issues) or $24 for three years (18 issues). Send order and payment to the address below or call MasterCard, Visa or American Express orders to our toll-free number.

Subscriptions in the UK are mailed by air and cost £10 for one year, £18 for two years or £25 for three years. Send order and payment to the UK address below.

Canadian and other foreign surface subscriptions cost $15 per year or $27 for two years and should be sent to the USA address.

We guarantee your satisfaction or we will refund your entire subscription price.

Needless to say, we can't fill up all the pages without your help. So send in your programs, articles, hints and tips. Remember, illustrations and screen photos make a piece much more interesting. Send in your reviews of peripherals and software too—but be warned: reviews must be in-depth and objective. We want you to respect what you read on the pages of **SYNC** so be honest and forthright in the material you send us. Of course we pay for contributions—just don't expect to retire on it.

The exploration has begun. Join us.

The magazine for Sinclair ZX80 users

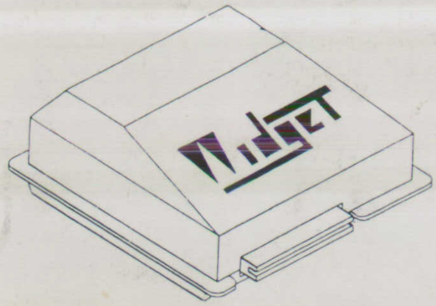
SYNC

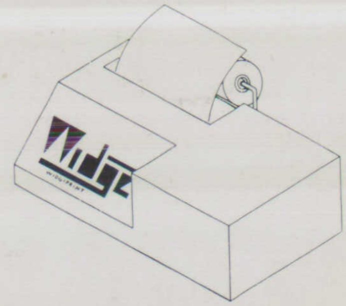
39 East Hanover Avenue
Morris Plains, NJ 07950, USA
Toll free **800-631-8112**
(In NJ 201-540-0445)

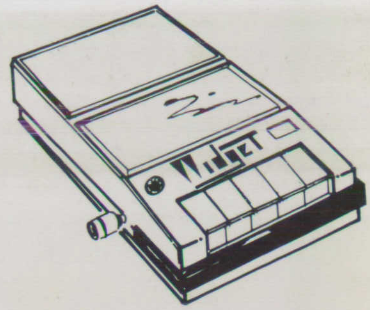
27 Andrew Close, Stoke Golding
Nuneaton CV13 6EL, England

THE CAI 'WIDGET'® SERIES OF PERIPHERALS FOR THE ZX-80 OPENS THE DOOR TO REAL COMPUTING

✔ *Check Our Specifications*







WIDGET® $49.95

PARALLEL TTL INPUT PORT

- Eight bits non-latched
- Draws one low power Schottky load
- Uses:
  - *Interface to user designed circuits*
  - *Sensing alarm conditions*
  - *Interpreting limit switches*
  - *Accepting user switches or other input devices*

PARALLEL TTL OUTPUT PORT

- Eight bits concurrently latched
- Supplies up to ten (10) low power Schottky loads
- Uses:
  - *Interface to user designed circuits*
  - *Activating Led's, Lamps and Indicators*
  - *Powering relays, SCRS and TRIACS for high current 110 VAC loads*
  - *Sounding alarms*

SERIAL RS-232 INPUT/OUTPUT PORT

- Three wire bi-directional serial line
- Baud rates strappable at 110, 300, 1200, 2400, 9600
- Communicate with:
  - *Serial printers*
  - *Modems (transmit and receive data only)*
  - *Terminals (teletypes and CRT's)*
  - *Graphic plotters*

ADDRESS AND CONTROL SIGNALS from the ZX-80 are all buffered to allow ease of expansion.

WIDGET INTERFACES DIRECTLY TO THE WIDGIPRINT and WIDGITAPE and the 16K RAM and 8K ROM by simply, plugging one into another.

The WIDGET, like all CAI devices, comes with complete schematics and instructions.

WIDGIPRINT® $99.95

HIGH QUALITY THERMAL PRINTER at a very affordable price

THE WIDGIPRINT PROVIDES THIRTY-TWO (32) COLUMNS OF OUTPUT in compliance with the format shown on the television screen

EASY INSTALLATION

- The CAI Printer is designed to plug directly into its own port on the Widget board. This leaves all eight I/O bits on the WIDGET board available to the user. There is no soldering or modifications necessary
- The printer is equipped with its own power supply

SPECIFIC PRINTER SPECIFICATIONS

- Full alpha-numeric capabilities with limited graphics
- Thirty-two (32) characters per line
- Self-contained power system
- Two-hundred lines / minute print speed
- Long life — few moving parts
- Paper size 3.15" roll style readily available • Size 6.4" x 1.9" x 3.7"

The PRINTER comes with a complete instruction and operations manual.

WIDGITAPE® $99.95

The WIDGITAPE is designed as a ***highly reliable*** mass storage device for the data generated with the ZX-80. This device interfaces through the CAI WIDGET Board, and is completely independent of the tape connections on the ZX-80.

The WIDGITAPE provides much of the functionality found in floppy disks for a fraction of the cost.

THIS INCLUDES

- Writing a file
- Reading a file
- Extending or modifying a file
- Deleting a file
- Each tape also maintains its own file directory which can be viewed on the television screen

All these functions are programmatically accessable to the user via simple keyboard commands.

INSTALLATION

- The WIDGITAPE plugs into its own plug on the WIDGET Board with no need to solder or modify the ZX-80.

USE THIS FORM TO ORDER YOUR WIDGET® PERIPHERALS

Send Check, Money Order Or Credit Card Number To:

Quantity	Description	Unit Price	Total
	WIDGET®	$49.95	
	WIDGIPRINT®	$99.95	
	WIDGITAPE®	$99.95	
	TOTAL		

CAI Instruments, Inc.
P.O. Box 446
Midland, MI 48640
517/835-6145

Add $5.00 for shipping and handling

Michigan residents add 4% sales tax

Mastercharge ____________ Visa ____________

Exp. Date ____________

Name ____________

Address ____________

City ____________ State ____________ ZIP ____________